A Tutorial on Advanced Data Visualization in R

Mehdi Ghaniabadi 11253463, Jingyi Xiao 11273254

Please note that in order to run the code properly, one should use RStudio, install all the packages (i.e. libraries) given in the beginning of the code, and change the data file path to where the data file is in one's computer. Afterwards, in order to run the code and build the dashboard, in RStudio, one should go to: Knit –> Knit to flex_dashboard.

1. Presentation of the subject

Data visualization is used to present a graphical view of the datasets. Each dataset is composed of different variables where their distributions and their underlying relationships are not apparent, especially when there is a large amount of data. Data visualization helps discover the shape of the data and how such variables interact with one another. Therefore, it enables us to understand the data better and make more informed decisions accordingly.

In order to make the most of data visualization on a specific dataset, one should try different graphs and see which can explain the data better and which are more appropriate for the task of interest. Therefore, in this tutorial, we present different types of graphs which are relevant in almost any types of datasets encountered in business contexts.

This tutorial makes use of the programming language R as an efficient tool to perform advanced data visualization. What makes this tutorial "advanced" as opposed to basic data visualization, is that it is interactive; it creates a dashboard which is nicely presentable by html format which can be published online; there are a variety of important interactive graphs inside the dashboard which are relevant in any dataset, such as, Bar Charts, Pie Charts, 3D Graphs, Maps, Tables, Pivot tables, and Gauges; and we use a very large real dataset which contains missing data, in order to show the full potential of visualization in R in practice. The resulting dashboard is also quite practical and can be used for data visualization in different business organizations, and enhances our understanding of a dataset significantly. It is worth mentioning that the dataset that we use is new and such visualizations are not implemented on that before.

The rest of this tutorial is structured as follows. In the next section a brief review of the literature is presented. In section 3, we describe different methods used in data visualization including dashboards and various types of graphs. Section 4 outlines different R resources available for data visualization. Finally, in section 5, the details on how to code dashboards and graphs in R is presented using a real dataset, the US Real Estate Data.

2. Literature review

As C Chen et al.(2008) mentioned in the Handbook of Data Visualization, although graphics have been widely utilized in statistics and science, there is not a standardized theory about data visualization. Besides, data visualization is an effcient and effective way to communicate and interpret information. So, it can be expected that the number of theoretical articles about data visualization is going to flourish in this day and age. The literature review is divided in two parts. First the references for data visualization in general is reviewed. Then we review the literature for data visualization in R.

2.1. General Data Visualization

C Chen, WK Härdle, A Unwin (2008) summarized a detailed collection of principles from different contributors, among which Michael Friendly (2008) outlined a history of data visualization. From 1800 to 1850, the modern graphics stepped into the vision of statisticians, based on the privious invention of bar charts, pie charts and various plots. In this book, the three editors presented a systematic and theoretical overview to data visualization, such as distinguishing the difference between presentation graphics and exploratory graphics. The formers which are generally static and of high quality, might not give clue to how to reach the conculsion, while the latters are used to seek results and should be informative. Their systematic summarization provides a solid foundation for the development of data visualization.

Data visualization gradually developes into varying ramifications according to the specific area it applies to. For example, Maniyar, D. M. et al (2006) performed feature selection by leveraging data visualization; Gorban, A. N. et al. (2008) applied data visualization into dimension reduction; Ramachandran, P., & Varoquaux, G. (2011) focused on 3D data visualization for scientific data.

MO Ward, G Grinstein, D Keim (2010) further provide a theory and application on interactive data visualization.

Knaflic, C. N. (2015) is a book designed to instruct data visualization for business professionals and is especially suitable for people who have not had much experience in the field.

Wilke (2019) can be considered as a reference manual for practical data visualization which also takes into account the aesthetic techniques for visualizing data along with its statistical aspects. The free online version of the book can be found in https://serialmentor.com/dataviz/.

The website https://www.data-to-viz.com/ is also a good online resource which provides a nice classification of various types of data visualization methods in order to find the most suitable ones based on the type of data and variables we have.

2.2. Data Visualization in R

Kabacoff, R. (2018) can be considered as a recent reference manual for data visualization techniques in R. Along with R codes of various graphs, it also provides a well-structured classification of data visualization methods and their description, based on the type of variables and data we have. This book is available online for free at https://rkabacoff.github.io/datavis/datavis.pdf.

The website DataCamp provides a series of five online courses dedicated for data visualization with R. The online courses are available at https://learn.datacamp.com/skill-tracks/data-visualization-with-r.

The website https://www.r-graph-gallery.com is a very practical and useful source for various types of graphs, maps, networks, time series and animation along with several examples and their R codes.

The Youtube website also provides different tutorials for data visualization with R. We specifically took advantage of the video https://www.youtube.com/watch?v=_a4S4tq62OE which provides a tutorial on interactive dashboard. Our work also was built upon what is done in this video. However, we modified the graphs and customized them based on our new dataset which contained big real data and also improved the aesthetic aspects of the work significantly. We also added other advanced graphs such as 3D scatter and mesh plots.

3. Description of the methods

In this part, we describe different methods used in data visualization including dashboards and various types of graphs. We also relate them to our real dataset and the graphs we have created in our dashboard, in order to make them clearer. For some variations of the methods presented in this section, we have not created a graph in our dashboard, in order not to make the tutorial too exhaustive. Besides, some graphs such as time series are not relevant to our data. For such cases, in the text we refer to the corresponding online sources to illustrate the graphs with examples and their R codes.

3.1. Dashboards

A data dashboard is a tool to manage information which tracks the situation of a company or an organization in a visual way by presenting the related data in charts, tables and gauges. They can be customized according to a specific task and can monitor the processes in real-time. A more comprehensive definition can be found in https://www.klipfolio.com/resources/articles/what-is-data-dashboard. Moreover, various real examples of data dashboards can be seen in this link: https://marketingplatform.google.com/about/data-studio/gallery/.

3.2. Univariate graphs

In univariate graphs, one wants to visualize a single variable, which corresponds to a single column in our dataset. We have two types of such graphs: categorical and numerical. It is better for the readers to run in R Markdown our flex dashboard first to knit a website interactive flex dashboard. This will give a more straight-forward visional perception of the following explanation.

3.2.1. Categorical:

In a univariate categorical graph, we have a single categorical variable. The column *"state_ab"* is an example of a categorical variable in our dataset.

The most common graphs to visualize such a variable are bar charts and pie charts.

In our data dashboard, the graph *"Regions surveyed By State"* is a bar chart for a categorical variable, which counts the number of times each category (here each state) is repeated in a single column (here the column *"state_ab"*). In other words, it counts how many observations of each category exists for a single variable. In our dataset, this counts for each state which is equivalent to the number of regions surveyed.

Moreover, the graph *"Top States of participating in the survey"* is a pie chart for a categorical variable, which shows the count of each category of a categorical variable in terms of percentage. You may attention to filter option for considering only top states with the highest number of regions surveyed. In this example, the pie chart shows the percentage number of regions surveyed for each state.

3.2.1. Numerical:

In a univariate numerical graph, we have a single numerical variable. The column *"rent_mean"* is an example of a numercial variable in our dataset. The most common graphs to visualize such a variable are histograms and density plots.

To build a histogram, the values of the variable is divided into several intervals then the number of observations in each interval is counted. Several examples of a histogram with their R codes can be found in https://www.r-graph-gallery.com/histogram.

The density plot shows the distribution, i.e., the density of the numerical variable. Several examples of a density plot with their R codes can be found in https://www.r-graph-gallery.com/density-plot.

3.3. Bivariate graphs

Bivariate graphs present the relationship between two variables which can be a combination of categorical or numerical variables, i.e., numerical vs numerical, categorical vs numerical, and categorical vs categorical.

For two numerical variables, scatterplots are common to be used. In our dashboard the graph "Scatter Plot of Percentage Married Vs Average household income" is an example of a scatterplot, which shows that there is a slight positive correlation between being married and the household income.

For one numerical and one categorical variables, bar charts and boxplots are common. In our dashboard, the graph *"Population surveyed in each state"* is a bar chart which considers two columns of the data, a categorical variable (*State*) and a numerical real variable (*Population*). Note that it stacks up the population of all regions inside a state. We can also have a pie chart based on these two variables where each portion of the pie chart may present the population percentage of each state in the united states. Moreover, the graph *"Box Plot of Average household income for each state"* is an example of boxplot for visualizing categorical vs numerical variables. More information on boxplots can be found in https://www.r-graph-gallery.com/boxplot. Ridgeline charts are also used to show the distribution of a numerical variable for different values of a category. Examples and R codes of ridgeline charts can be found in https://www.r-graph-gallery.com/ridgeline-plot. Violin charts can also be used for the same purpose but with a different type of visualization (https://www.r-graph-gallery.com/violin.html).

For two categorical variables, Grouped and Stacked bar charts are used. We refer to the link https://rkabacoff.github.io/datavis/Bivariate.html#Categorical-Categorical for a detailed study of such charts and their R codes.

3.4. Multivariate graphs

Multivariate graphs are used to demonstrate the relationship between three or more variables, for any combination of categorical or numerical variables. In our dashboard the *3D scatterplots and 3D mesh plots* visualize the relationship between three numerical variables. Grouping and faceting are two other methods for visualizing multivariate graphs which are presented in https://rkabacoff.github.io/datavis/Multivariate.html along with their R codes.

3.5. Maps

Maps are used to visualize data-driven information regarding different regions of a geographical location. There are several methods to visualize maps. A choropleth map is used to give information regarding a numerical variable in different regions of a map, where the intensity of color of each region corresponds to the value of that numerical variable in that region. In our dashboard, the tab *"Map"* actually presents a choropleth map where the states with a higher number of surveyed regions have a more intense color.

A connection map shows the connection different regions to each other. Examples and R codes of connection maps can be found in https://www.r-graph-gallery.com/connection-map.

A bubble map is also a popular type of map visualization where the size or color intensity of circles demonstrate the value of a numerical variable in different region. The corresponding examples of such maps and their R codes can be found in https://www.r-graph-gallery.com/bubble-map.

3.6. Pivot table

A pivot table is used to present the relationship between different variables using a variety of predetermined graphs such as bar chart, scatter chart, heatmap, treemap, line chart, area chart, etc. It also summarizes the data using different statistics such as average, count, median, variance, etc. More details will be given in section 5 where a pivot table is applied to our dataset.

3.7. Other visualization methods

Time series are also an important area of data science which have special characteristics which makes their visualization unique compared to the methods described above. Various examples of time series and their R codes can be found in https://www.r-graph-gallery.com/time-series.html.

Animated graphs are also an advanced method for data visualization which present different states of a graph in a sequential and dynamic way. The gganimate library in R is used to build such graphs. Examples and R codes of animated graphs are given in https://www.r-graph-gallery.com/animation.html.

The website https://www.r-graph-gallery.com contains a variety of other graphs along with their R codes which would be too exhaustive for this tutorial to present.

4. R resources

In our implementation of data visualization presented in section 5, we have used twelve R libraries **including flexdashboard, shinythemes, Hmisc, knitr, DT, rpivotTable, ggplot2, plotly, dplyr, openintro, highcharter, and ggvis**. The R documentations for each of the libraries related to data visualization used in this project are presented in the appendix 7.1. The examples related to such libraries for visualizing our dataset are given in section 5.

Apart from the libraries we use in section 5, there are many other extremely powerful libraries designed for data visualization, and those libraries will be introduced in section 4.2. Due to the fact that this report is not meant to be too extensive, we do not apply each of those nicely visual libraries in our project. Data visualization is used in practice frequently; therefore, we will keep exploring such libraries in the future.

4.1 R resources used in this project

This table summarizes the information about **6 main interactive libraries** we use extensively in section 5.

Package	category	Description	Authors	Manual
flexdashboard	data visualization	converts an R Markdown document to web flexible dashboard	Richard Iannone	https://cran.r-project.org/web/packages/flexdashboard/flexdashboard.pdf
shiny	data visualization	builds interactive web applications with R	Winston Chang et al.	https://cran.r-project.org/web/packages/shiny/shiny.pdf
plotly	data visualization	creates interactive web graphics from ‘ggplot2’ graphs	Carson Siever et al.	https://cran.r-project.org/web/packages/plotly/plotly.pdf
DataTable	data visualization	wrapper of the JavaScript library ‘DataTables’ to create web data table	Yihui Xie	https://cran.r-project.org/web/packages/DT/DT.pdf
rpivotTable	data visualization	build pivot table and dynamically slice & dice the data	Enzo Martoglio et al.	https://cran.r-project.org/web/packages/rpivotTable/rpivotTable.pdf

Package	category	Description	Authors	Manual
highcharter	data visualization	charting library offering numerous chart types	Joshua Kunst	https://cran.r-project.org/web/packages/highcharter/highcharter.pdf

4.2 R resources for general data visualization

The following list is a summary of **22 other available libraries in R** to perform data visualization. This list is derived mainly from 3 resourses: Machlis Musings (2019) and Kabacoff, R. (2018), and CRAN. We mention the website of Kabacoff, R. (2018) in the previous section, one can also visit resources of Machlis Musings in (https://www.computerworld.com/article/2921176/great-r-packages-for-data-import-wrangling-visualization.html).

Package	category	Description	Authors	Manual
patchwork	data visualization	expands the API compared to 'ggplot2'to allow for arbitrarily complex composition; provide mathematical operators for combining multiple plots	Thomas Lin Pedersen	https://cran.r-project.org/web/packages/patchwork/patchwork.pdf
ggforce	data visualization	strengthen the ability of ggplot2 on composing specilialised plots	Thomas Lin Pedersen	https://cran.r-project.org/web/packages/ggforce/ggforce.pdf
esquisse	data visualization	A 'shiny' gadget that interactively explore and visualize data with 'ggplot2' charts	Fanny Meyer, Victor Perrier, Ian Carroll	https://cran.r-project.org/web/packages/esquisse/esquisse.pdf
dygraphs	data visualization	interactive time series charting library	Dan Vanderkam et al.	https://cran.r-project.org/web/packages/dygraphs/dygraphs.pdf
metricsgraphics	data visualization	create interactive charts with the JavaScript library	Bob Rudis et al.	https://cran.r-project.org/web/packages/metricsgraphics/metricsgraphics.pdf

Package	category	Description	Authors	Manual
echarts4r	data visualization	A html library that creates interactive graphics by leveraging the 'Echarts Javascript' library which includes 34 chart types, themes, 'Shiny' proxies and animations	John Coene	https://cran.r-project.org/web/packages/echarts4r/echarts4r.pdf
rbokeh	data visualization	create interactive web-based graphics	Ryan Hafen	http://hafen.github.io/rbokeh/; https://cran.r-project.org/web/packages/rbokeh/rbokeh.pdf
rCharts	data visualization	create interactive javascript visualizations	Ramnath Vaidyanathan	https://github.com/ramnathv/rCharts
RColorBrewer	data visualization, color	contains a ready-to-use color palettes for creating beautiful graphics	Erich Neuwirth	https://cran.r-project.org/web/packages/RColorBrewer/RColorBrewer.pdf
paletteer	data visualization, color	collections of color palettes	Emil Hvitfeldt et al.	https://cran.r-project.org/web/packages/paletteer/paletteer.pdf
colourpicker	data visualization, color	color picker tool for shiny and other plots	Dean Attali, David Griswold	https://cran.r-project.org/web/packages/colourpicker/colourpicker.pdf
rcdimple	data visualization	provides easy access to the dimple d3.js plotting library in the form of an htmlwidget	Kent Russell et al.	https://www.rdocumentation.org/packages/rcdimple
taucharts	data visualization	the html widget library useful for scatterplots	Bob Rudis	https://rpubs.com/hrbrmstr/taucharts
quantmod	financial data visualization, data import, data analysis	analyse quantitative financial trading strategies	Jeffrey A. Ryan et al.	https://cran.r-project.org/web/packages/quantmod/quantmod.pdf

Package	category	Description	Authors	Manual
tidyquant	financial data visualization	quantitative financial analysis library	Matt Dancho, Davis Vaughan	https://cran.r-project.org/web/packages/tidyquant/tidyquant.pdf
geofacet	geographical data visualization, mapping	provides geofaceting functionality for 'ggplot2' in terms of geographical data	Ryan Hafen, Barret Schloerke	https://cran.r-project.org/web/packages/geofacet/geofacet.pdf
sf	mapping	operates geospatial data	Edzer Pebesma et al.	https://cran.r-project.org/web/packages/sf/sf.pdf
mapsapi	mapping	interface to 'Google Map' APIs	Michael Dorman, et al.	https://cran.r-project.org/web/packages/mapsapi/mapsapi.pdf
tidycensus	mapping	interface to the decennial US Census and American Community Survey APIs and the US Census Bureau's geographic boundary files.	Kyle Walker, et al.	https://cran.r-project.org/web/packages/tidycensus/tidycensus.pdf
leaflet	mapping	creates Interactive Web Maps with the JavaScript 'Leaflet' library	Joe Cheng et al.	https://cran.r-project.org/web/packages/leaflet/leaflet.pdf
ggmap	mapping	visualize spatial data with 'ggplot2', and can models based on various online resources such as 'Google Map'	David Kahle et al.	https://cran.r-project.org/web/packages/ggmap/ggmap.pdf
rgeocodio	mapping	perform geocoding	Bob Rudis	https://github.com/hrbrmstr/rgeocodio
tmap & tmaptools	mapping	create thematic maps	Martijn Tennekes	https://cran.r-project.org/web/packages/tmaptools/tmaptools.pdf

5. Graphs analyses with the US real estate data

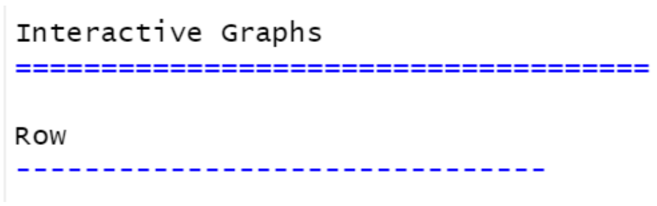
In this section, we will apply data visualization to create a dashboard with the United States Real Estate data. This dataset surveys 39,000 unique regions over 50 states in the United States, specifying in each region the average rent, rent burden indicator (*rent_gt_40*), average household income, marriage rate, etc. The relationships among different variables will be examined using various graphs. The following table details the covariates in our dataset which are used in depicting the graphs.

Variable	Explanation
state_ab	State abbreviation in the US
pop	The population of the specified region in one state
rent_mean	The mean gross rent of the specified region
rent_gt_40	The percentage of population in a specified region that housing rent of an individual will be greater than 40% of his household income in the past 12 months
hi_mean	The mean household income of the specified region
married	The marriage rate of the specified region
hi_meanround	The 2-decimal rounded value of hi_mean
rent_gt_40round	The 1-decimal rounded value of rent_gt_40
marriedround	The 1-decimal rounded value of married

First, we are supposed to **install packages flexdashboard, shinythemes, highcharter, DT, rpivotTable, plotly, Hmisc, knitr, ggplot2, dplyr, openintro, ggvis** in order for the whole flex dashboard to run smoothly. Flex dashboard is produced under R markdown environment. To initialize a new flex dashboard, one may go to "file -> new file -> R markdown -> From template -> Flex Dashboard".

In order to run our code, one should first open the file of our code using RStudio, install the packages, and also change the **file path of read.csv**. Afterwards, in order to run the code and build the dashboard, in RStudio, one should go to: Knit –> Knit to flex_dashboard.

In the dashboarding, we can create new tabs by adding a seires of *equality sign "==="*; within one tab, different rows or columns can also be designed by adding a series of *dash line "- - -"* and specifying rows or columns above the dash lines in Figure 1.



```
Interactive Graphs
=====================================

Row
-------------------------------------
```

Figure 1: Adding new tabs and rows/columns

The dashboard file has been attached with this report, which can be knitted into an HTML file and is shareable to Twitter or FaceBook. The following are showcases of our project dashboard. In the next parts, graphs in each module, as shown in Figures 2, 3, 4 and 5, will be introduced and explained separately.

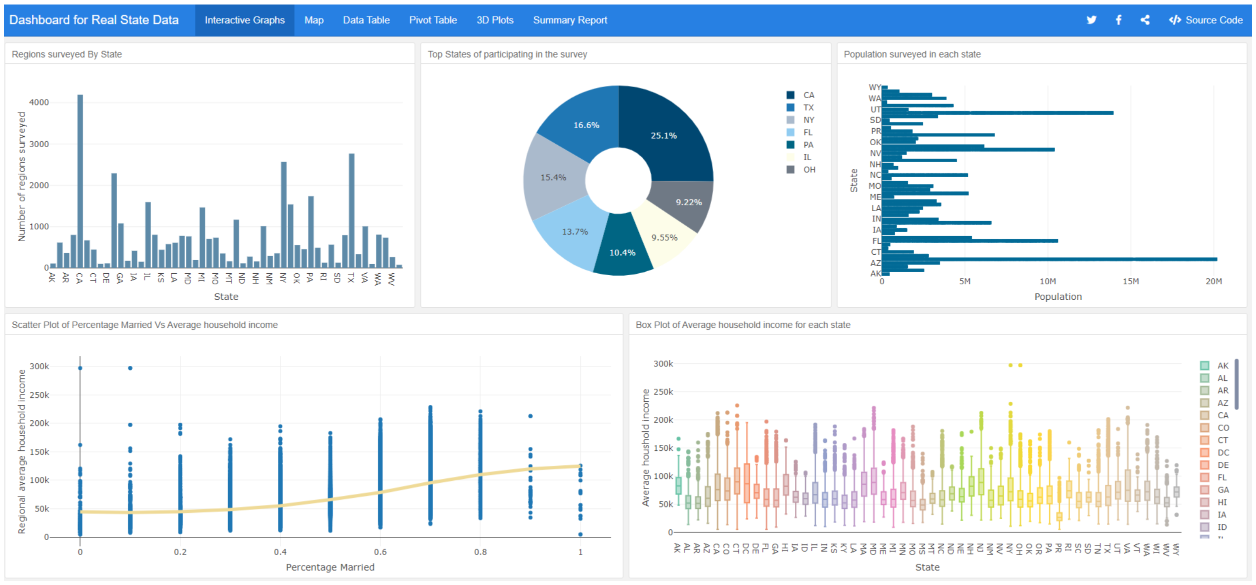
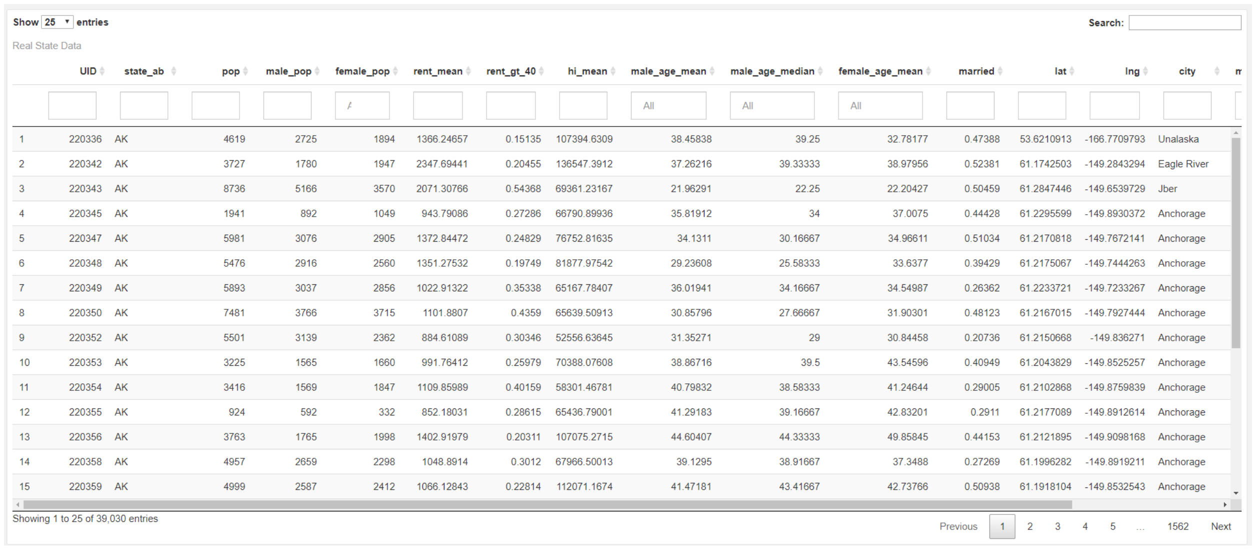


Figure 2: Dashboarding: Interactive graphs



Show 25 entries

Search:

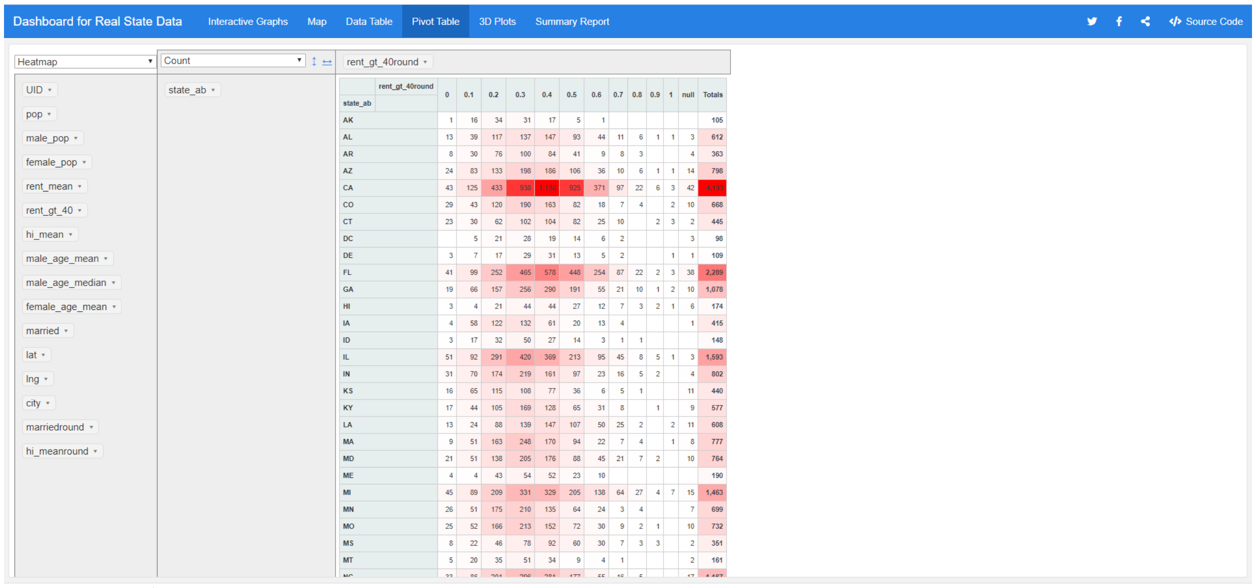
Real State Data

	UID	state_ab	pop	male_pop	female_pop	rent_mean	rent_gt_40	hi_mean	male_age_mean	male_age_median	female_age_mean	married	lat	lng	city
									All	All	All				
1	220336	AK	4619	2725	1894	1366.24657	0.15135	107394.6309	38.45838	39.25	32.78177	0.47388	53.6210913	-166.7709793	Unalaska
2	220342	AK	3727	1780	1947	2347.69441	0.20455	136547.3912	37.26216	39.33333	38.97956	0.52381	61.1742503	-149.2843294	Eagle River
3	220343	AK	8736	5166	3570	2071.30766	0.54368	69361.23167	21.96291	22.25	22.20427	0.50459	61.2847446	-149.6539729	Jber
4	220345	AK	1941	892	1049	943.79086	0.27286	66790.89936	35.81912	34	37.0075	0.44428	61.2295599	-149.8930372	Anchorage
5	220347	AK	5981	3076	2905	1372.84472	0.24829	76752.81635	34.1311	30.16667	34.96611	0.51034	61.2170818	-149.7672141	Anchorage
6	220348	AK	5476	2916	2560	1351.27532	0.19749	81877.97542	29.23608	25.58333	33.6377	0.39429	61.2175067	-149.7444263	Anchorage
7	220349	AK	5893	3037	2856	1022.91322	0.35338	65167.78407	36.01941	34.16667	34.54987	0.26362	61.2233721	-149.7233267	Anchorage
8	220350	AK	7481	3766	3715	1101.8807	0.4359	65639.50913	30.85796	27.66667	31.90301	0.48123	61.2167015	-149.7927444	Anchorage
9	220352	AK	5501	3139	2362	884.61089	0.30346	52556.63645	31.35271	29	30.84458	0.20736	61.2150668	-149.836271	Anchorage
10	220353	AK	3225	1565	1660	991.76412	0.25979	70388.07608	38.86716	39.5	43.54596	0.40949	61.2043829	-149.8525257	Anchorage
11	220354	AK	3416	1569	1847	1109.85989	0.40159	58301.46781	40.79832	38.58333	41.24644	0.29005	61.2102868	-149.8759839	Anchorage
12	220355	AK	924	592	332	852.18031	0.28615	65436.79001	41.29183	39.16667	42.83201	0.2911	61.2177089	-149.8912614	Anchorage
13	220356	AK	3763	1765	1998	1402.91979	0.20311	107075.2715	44.60407	44.33333	49.85845	0.44153	61.2121895	-149.9098168	Anchorage
14	220358	AK	4957	2659	2298	1048.8914	0.3012	67966.50013	39.1295	38.91667	37.3488	0.27269	61.1996282	-149.8919211	Anchorage
15	220359	AK	4999	2587	2412	1066.12843	0.22814	112071.1674	41.47181	43.41667	42.73766	0.50938	61.1918104	-149.8532543	Anchorage

Showing 1 to 25 of 39,030 entries

Previous 1 2 3 4 5 … 1562 Next

Figure 3: Dashboarding: Data table



state_ab \ rent_gt_40round	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	null	Totals
AK	1	16	34	31	17	5	1						105
AL	13	39	117	137	147	93	44	11	6	1	1	3	612
AR	8	30	76	100	84	41	9	8	3			4	363
AZ	24	83	133	198	186	106	36	10	6	1	1	14	798
CA	43	125	433	938	1,188	925	371	97	22	6	3	42	4,193
CO	29	43	120	190	163	82	18	7	4		2	10	668
CT	23	30	62	102	104	82	25	10		2	3	2	445
DC		5	21	28	19	14	6	2				3	98
DE	3	7	17	29	31	13	5	2			1	1	109
FL	41	99	252	465	578	448	254	87	22	2	3	38	2,289
GA	19	66	157	256	290	191	55	21	10	1	2	10	1,078
HI	3	4	21	44	44	27	12	7	3	2	1	6	174
IA	4	58	122	132	61	20	13	4				1	415
ID	3	17	32	50	27	14	3	1	1				148
IL	51	92	291	420	369	213	95	45	8	5	1	3	1,593
IN	31	70	174	219	161	97	23	16	5	2		4	802
KS	16	65	115	108	77	36	6	5	1			11	440
KY	17	44	105	169	128	65	31	8		1		9	577
LA	13	24	88	139	147	107	50	25	2		2	11	608
MA	9	51	163	248	170	94	22	7	4		1	8	777
MD	21	51	138	205	176	88	45	21	7	2		10	764
ME	4	4	43	54	52	23	10						190
MI	45	89	209	331	329	205	138	64	27	4	7	15	1,463
MN	26	51	175	210	135	64	24	3	4			7	699
MO	25	52	166	213	152	72	30	9	2	1		10	732
MS	8	22	46	78	92	60	30	7	3	3		2	351
MT	5	20	35	51	34	9	4	1				2	161

Figure 4: Dashboarding: Pivot table

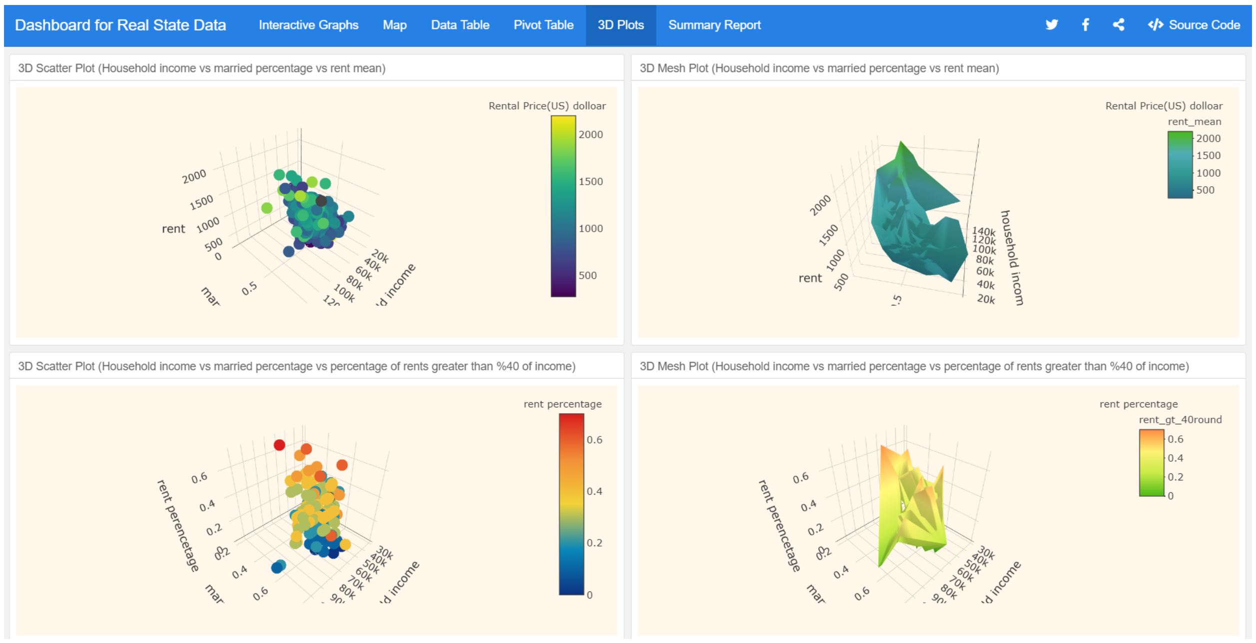

Dashboard for Real State Data
Interactive Graphs
Map
Data Table
Pivot Table
3D Plots
Summary Report
Source Code
3D Scatter Plot (Household income vs married percentage vs rent mean)
3D Mesh Plot (Household income vs married percentage vs rent mean)
Rental Price(US) dolloar
rent_mean
rent
household incom
3D Scatter Plot (Household income vs married percentage vs percentage of rents greater than %40 of income)
3D Mesh Plot (Household income vs married percentage vs percentage of rents greater than %40 of income)
rent percentage
rent_gt_40round
rent perencetage

Figure 5: Dashboarding: 3D plots

1) Interactive graphs

In our flexdashboard, **plotly** library is intensively used to draw interactive graphs. Other libraries, such as **leaflet**, **rbokeh**, **rCharts** and **highcharter**, can also return various attractive graphs. But in this dashboard, we will focus on **plotly**.

Bar chart

The first bar chart groups by different states and records the number of regions surveyed in each state. Two axes need to be input into **plotly**, and type is set to *bar*. It is noticeable that in **plotly** library, we do not code *"color ="* directly as the input. But rather, we need to use *"marker ="* to specify the list of color(s) in the graph. RGB or Hex are available color index in the library. Names of coordinates can also be added with *layout()*.

```r
p1 <- data %>%
         group_by(state_ab) %>%
         summarise(count = n()) %>%
         plot_ly(x = ~state_ab,
                 y = ~count,
# here we don't code it as color=... directly,
#but change it from color to marker=list(color=...)
                 marker = list(color = 'rgb(93,138,168)'),
                 type = 'bar') %>%
layout(xaxis = list(title = "State"),
yaxis = list(title = 'Number of regions surveyed'))
p1
```

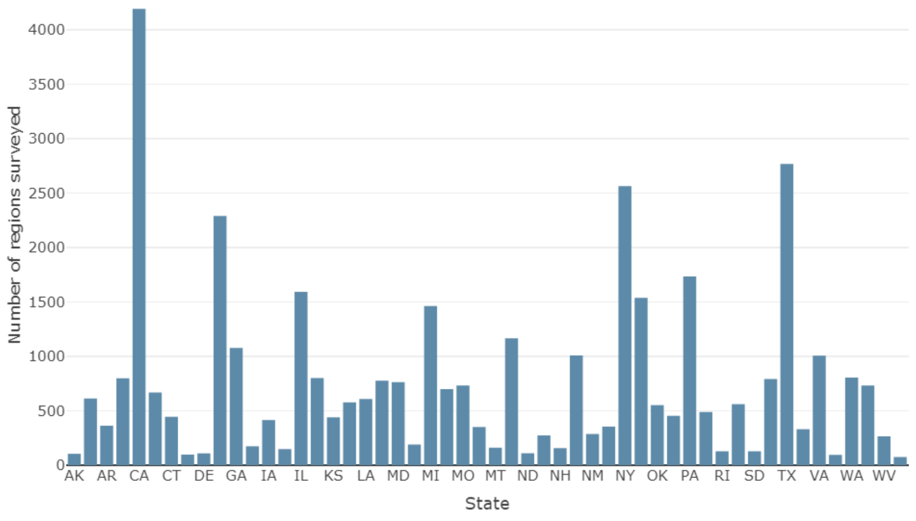


Figure 6: Bar chart

In this bar chart, it can be seen that regions in California, Texas, Florida, New York are the tops of places being surveyed.

Pie chart

The next graph is pie chart, which filters the most frequently surveyed states (numbers of surveyed regions larger than 1500). *filter* can be added after statistics are grouped by and counts. In the *marker*, we can also input a customized variable *mycolors*, a vector containing selected colors. To present only a clear pie chart, lines and grids in y and x axes are set to *FALSE*. Otherwise, coordinates and grids will be seen in the pie chart.

```
# self-defined color vector
mycolors <- c("#004771", "#91CCF1", "#FDFDE9", "#AABACC","#6F7985","#006583")
```

```
p2 <- data %>%
         group_by(state_ab) %>%
         summarise(count = n()) %>%
   #use filter to filter out states with
   #number of survey regions larger than 1500
         filter(count>1500) %>%
         plot_ly(labels = ~state_ab,
                 values = ~count,
                 marker = list(colors = mycolors)) %>%
   #add_pie belongs to function add_trace(),
   #Add trace(s) to a plotly visualization
   #add_pie to tune more specific parameters related to pie chart,
   #eg. the hole in the center
         add_pie(hole = 0.35) %>%
   #to present only a clear pie chart, we set all lines and grid = F
         layout(xaxis = list(zeroline = F,
                             showline = F,
                             showticklabels = F,
                             showgrid = F),
                yaxis = list(zeroline = F,
                             showline = F,
                             showticklabels=F,
                             showgrid=F))
p2
```

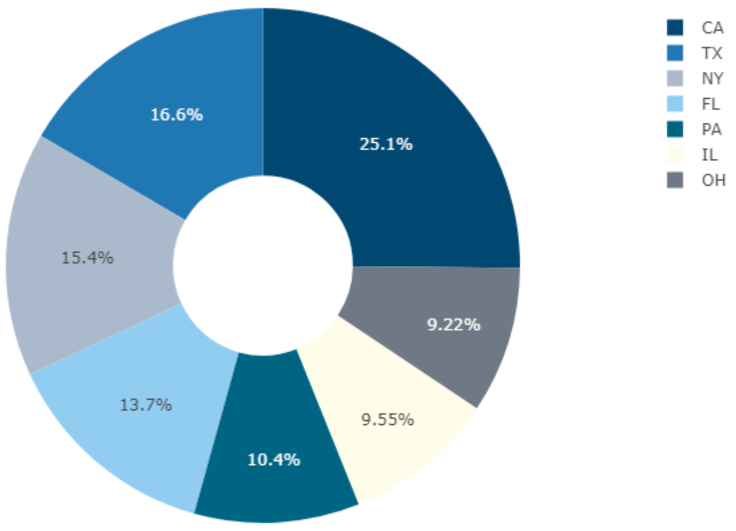


Figure 7: Pie chart

Turning to the pie chart itself, seven states are filtered with constraint 1500 surveyed regions. They are California, Texas, New York, Florida, Pennsylvania, Illinois and Ohio. Among regions in these seven states, California owns the maximum regions participating in this survey.

Horizontal bar chart

The third graph is another bar chart. The difference is that we will show bars depicted in a horizontal direction (*y axis*), by changing the *layout()*. Besides, we can use *text = paste()* to assign a customized name card for each observation, each surveyed region in this case. When mouse is placed on a specific observation, it will present an interactive card with customized information specified in *paste()*.

```
p3 <- plot_ly(data,
              x = ~pop,
              y = ~state_ab,
              marker = list(color = 'rgb(1,106,150)'),
              # use text function to add "name card" for each obs
              text = paste("State:", data$state_ab,
                           "Population:",
                           data$pop),
              type = "bar") %>%
         layout(yaxis = list(title="State"),
                xaxis = list(title = "Population"))
p3
```

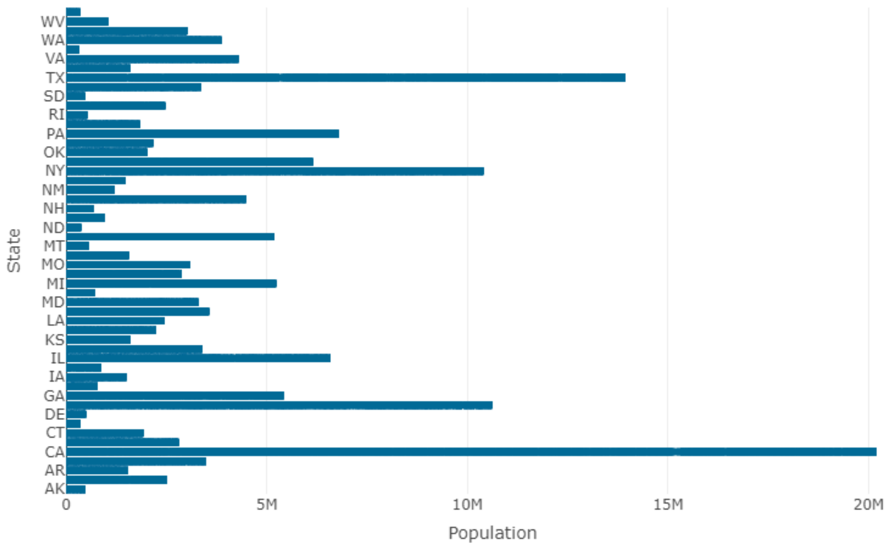


Figure 8: Horizontal bar chart

Similar to the first bar chart, California, Texas, New York, Florida own the largest population participating in this survey. The reason is that they are more populated and also more regions are surveyed in those states.

Scatter Plot with regression line

In the fourth graph, we depict the relationship between marriage ratio and average household income in each surveyed region. The dataset contains missing values in *hi_mean* and *married*, so we use *impute* function in package **Hmisc** to impute the missing values with the expectation of the covariates. There are also other methods dealing with missing value. We can eliminate the observations if they are not crucial to the analysis.

Besides, using regression to impute can have a more accurate value. Here, we use the expectation of the covariate to smooth those missing values.

In order to present the relationship in a more organized way, marriage ratio, originally ranging from 0 to 1, is now rounded to 1 digit, and household income is rounded to 2 digits. Plotting these two covariates is difficult to see the detail of their relationship. Therefore, a regression line is added to assist in depicting their relationship. In *add_lines* function, we can fit regression in a local neighborhood by loess approach, short for Local Regression. Furthermore, parameter span in *loess* function can be used to smoothen the regression curve. The greater the value of *span* we choose in *loess* function, the smoother the regression curve is.

```
data$hi_mean <- with(data, impute(data$hi_mean, mean))
data$married <- with(data, impute(data$married, mean))
data <- data %>%
  mutate(marriedround= round(married, digits = 1),hi_meanround=round(hi_mean,digits=2))

p4 <- plot_ly(data, x=~marriedround) %>%
         add_markers(y = ~hi_meanround,
                     text = ~paste("Average household income: ",
                                   hi_meanround,"married:", marriedround),
                     showlegend = F) %>%
                     add_lines(y = ~fitted(loess(hi_meanround ~
                                                  marriedround)),
                   name = "Loess Smoother",
                   color = I("#EFDA97"),
                   showlegend = T,
                   line = list(width=5)) %>%
         layout(xaxis = list(title = "Percentage Married"),
                yaxis = list(title = "Regional average household income"))
p4
```

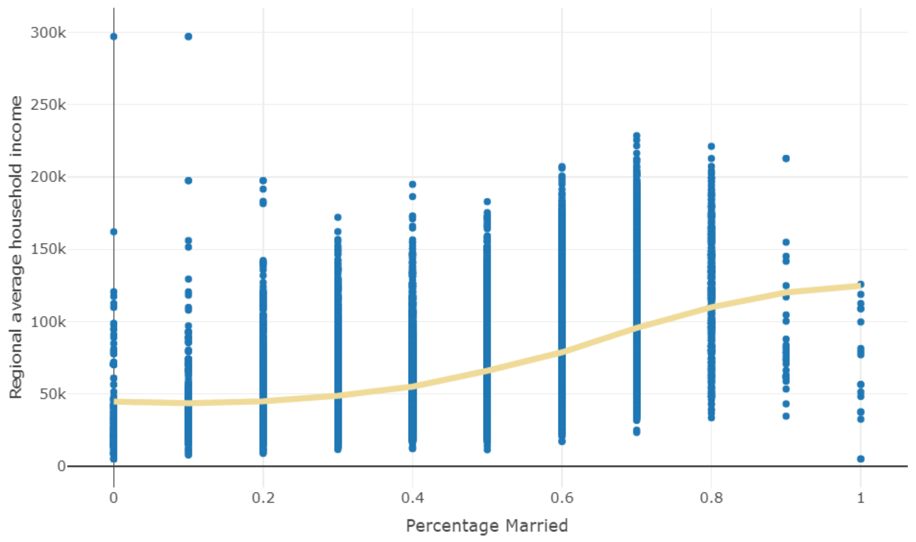


Figure 9: Scatter plot with regression line

Intuitively, higher marriage rate is usually followed by a higher average household income because there are more than one person contributing salaries in the family. While, the fitted curve of *loess* function further shows that as marriage rate increases, the marginal change in household income is higher when marriage rate is between 0.5 and 0.7.

Box Plot

Box plot in **plotly** library can tell us the *outlier* (in this case, it will be household income). Apart from value of each outlier, the interactive graph allows us to see the *minimum, first quantile, median, third quantile, upper fence and maximum* household income for each state.

```
p4.2 <- plot_ly(data, y = ~hi_meanround, color = ~state_ab, type =
                    "box")%>%
                layout(xaxis = list(title = "State"),
                yaxis = list(title = 'Average houshold income'))
p4.2
```

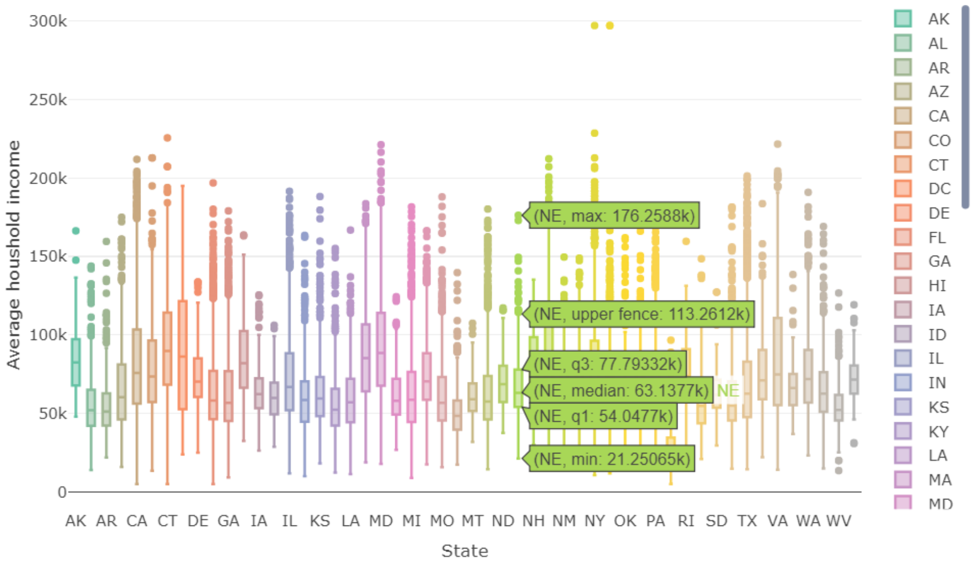


Figure 10: Box plot

2) Geographical map

In the second tab of the dashboarding, we utilize *highchart()* function to depict a US real estate map. *Highchart*, which comes from library **highcharter**, is a powerful function for drawing geographical details, and it can even draw as precise as countryside geographical information within a country. Apart from using *highchart()*, one can use directly *hcmap()* to chart a geographical map.

Maps can be selected from the highmaps collection https://code.highcharts.com/mapdata/ by a *url* address.

Going back to *highchart()* function, one can use *hc_title()* to add a general title to the map, *hc_subtitle()* to add another subtitle under the general title. The most interesting thing in *highchart()* is *hec_add_series_map*. By adding *usgeojson*, we can load the US map from a *geojson* map. We select state as *name*, total as *value* (*the name of column*) in p5 dataset. *woename* is used to join the map and p5 dataset. The color scale in *highmap* reflects the variety of our data. We might use *highchart()* series to examine other meaningful distribution, for example, average household income in each state.

```
#group by states to sum up number of regions being surveyed
p5 <- data %>%
         group_by(state_ab) %>%
         summarize(total = n())
# use abbr2state to modify the abbreivation to full name
#of each state in the US
p5$State <- abbr2state(p5$state_ab)

highchart() %>%
   #add general title
         hc_title(text = "Number of regions surveyed in US") %>%
   #add subtitle
         hc_subtitle(text = "Source: USA real estate survey") %>%
# differing map can be chosen from
         hc_add_series_map(usgeojson, p5,
                           name = "State",
                           #value: the name of the columnn in p5.
                           value = "total",

                           joinBy = c("woename", "State")) %>%
   # enable mapnavigation to allow zooming
         hc_mapNavigation(enabled = T)
```

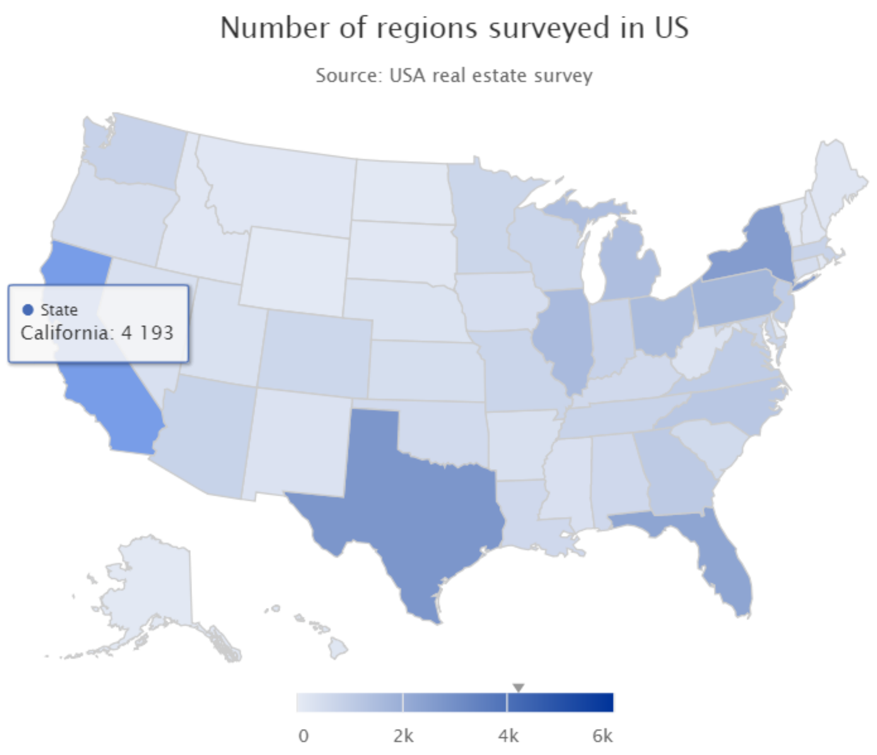
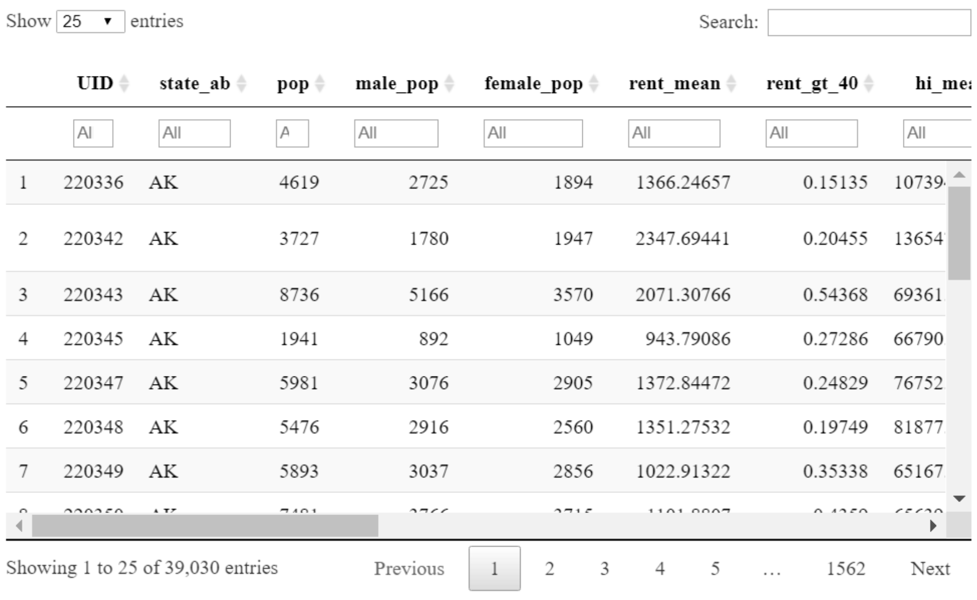

Number of regions surveyed in US
Source: USA real estate survey
● State
California: 4 193
0
2k
4k
6k

Figure 11: Geographical map

3) Data table

The following module is the data table, which can be easily coded but it is a very convenient way to transfer the original dataset into website or dashboard. The interface in the data table is user-friendly and interactive. The operation is similar to what can be achieved in spreadsheet. we can specify the position of *filter* to top or bottom in this data table. Furthermore, the *options* can control how many observations users can see in one page.

```r
datatable(data,
          caption = "Real State Data",
          # rownames=TRUE, showing the rownames in dataset
          rownames = T,
          # filter can be top or bottom
          filter = "top",
          options = list(pageLength = 25))
```



Show 25 entries   Search:

	UID	state_ab	pop	male_pop	female_pop	rent_mean	rent_gt_40	hi_me
	Al	All	A	All	All	All	All	All
1	220336	AK	4619	2725	1894	1366.24657	0.15135	10739
2	220342	AK	3727	1780	1947	2347.69441	0.20455	13654
3	220343	AK	8736	5166	3570	2071.30766	0.54368	69361
4	220345	AK	1941	892	1049	943.79086	0.27286	66790
5	220347	AK	5981	3076	2905	1372.84472	0.24829	76752
6	220348	AK	5476	2916	2560	1351.27532	0.19749	81877
7	220349	AK	5893	3037	2856	1022.91322	0.35338	65167

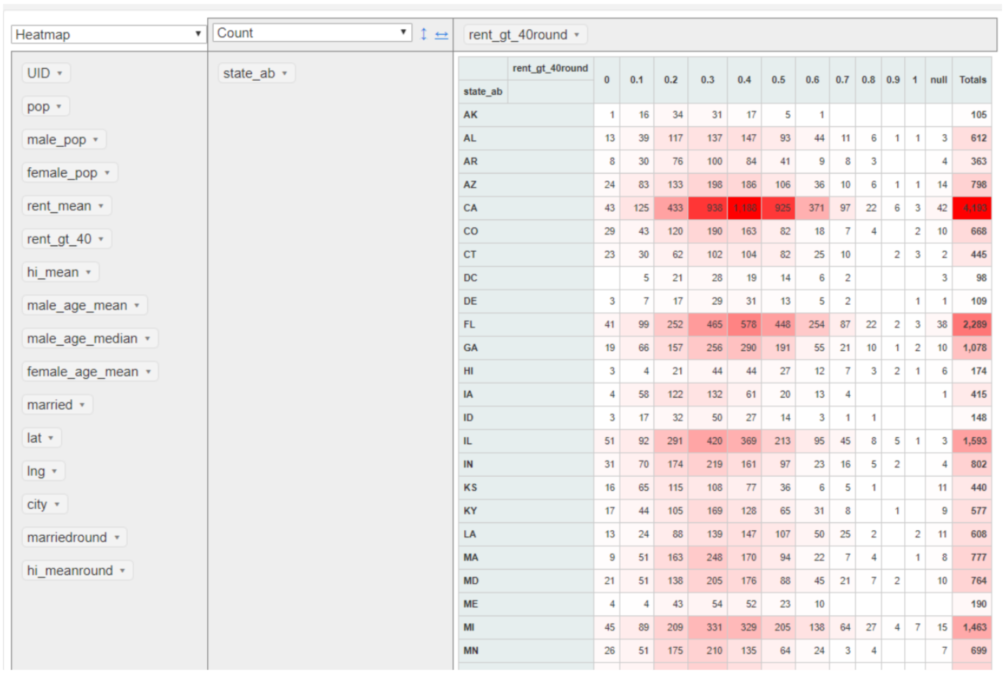
Showing 1 to 25 of 39,030 entries   Previous 1 2 3 4 5 … 1562 Next

Figure 12: Data table

4) Pivot table

Another useful table in dashboarding is pivot table. Here, heatmap is set to be an initializing graph when we first access to pivot table tab. *rpivotTable()* allows us to access to differing graphs, such heatmap, treemap, table barchart, scatter chart, area chart, etc. *rpivotTable* is the visualization version of the *datatable* described in the last section. In this table, we can change coordinates by adding or moving variables to y and x axes. Therefore, *rpivotTable* is very flexible and interpretable.

```r
data <- data%>%
  mutate(rent_gt_40round=round(rent_gt_40,digits=1))

rpivotTable(data,
            #string name to define how we like to statistize the heatmap
            aggregatorName = "Count",
            #column
            cols= "rent_gt_40round",
            rows = "state_ab",
            rendererName = "Heatmap")
```



rent_gt_40round / state_ab	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	null	Totals
AK	1	16	34	31	17	5	1						105
AL	13	39	117	137	147	93	44	11	6	1	1	3	612
AR	8	30	76	100	84	41	9	8	3			4	363
AZ	24	83	133	198	186	106	36	10	6	1	1	14	798
CA	43	125	433	938	1,188	925	371	97	22	6	3	42	4,193
CO	29	43	120	190	163	82	18	7	4		2	10	668
CT	23	30	62	102	104	82	25	10		2	3	2	445
DC		5	21	28	19	14	6	2				3	98
DE	3	7	17	29	31	13	5	2			1	1	109
FL	41	99	252	465	578	448	254	87	22	2	3	38	2,289
GA	19	66	157	256	290	191	55	21	10	1	2	10	1,078
HI	3	4	21	44	44	27	12	7	3	2	1	6	174
IA	4	58	122	132	61	20	13	4				1	415
ID	3	17	32	50	27	14	3	1	1				148
IL	51	92	291	420	369	213	95	45	8	5	1	3	1,593
IN	31	70	174	219	161	97	23	16	5	2		4	802
KS	16	65	115	108	77	36	6	5	1			11	440
KY	17	44	105	169	128	65	31	8		1		9	577
LA	13	24	88	139	147	107	50	25	2		2	11	608
MA	9	51	163	248	170	94	22	7	4		1	8	777
MD	21	51	138	205	176	88	45	21	7	2		10	764
ME	4	4	43	54	52	23	10						190
MI	45	89	209	331	329	205	138	64	27	4	7	15	1,463
MN	26	51	175	210	135	64	24	3	4			7	699

Figure 13: Pivot table – heat map

For example, we can switch to an area chart with the same column (*rent_gt_40round*) and row variable (*state_ab*).

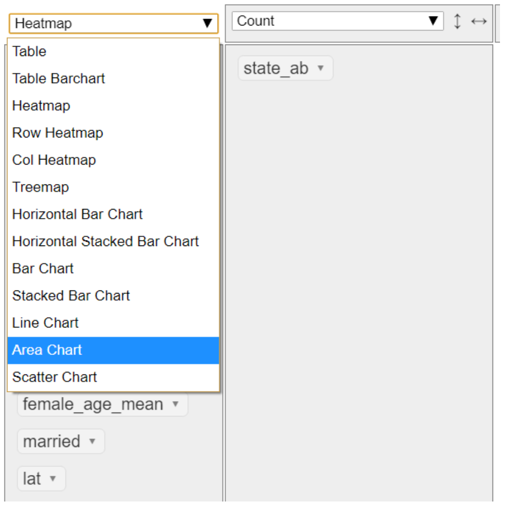


Figure 14: Pivot table – table options

Varying types of table can be chosen in the list above.

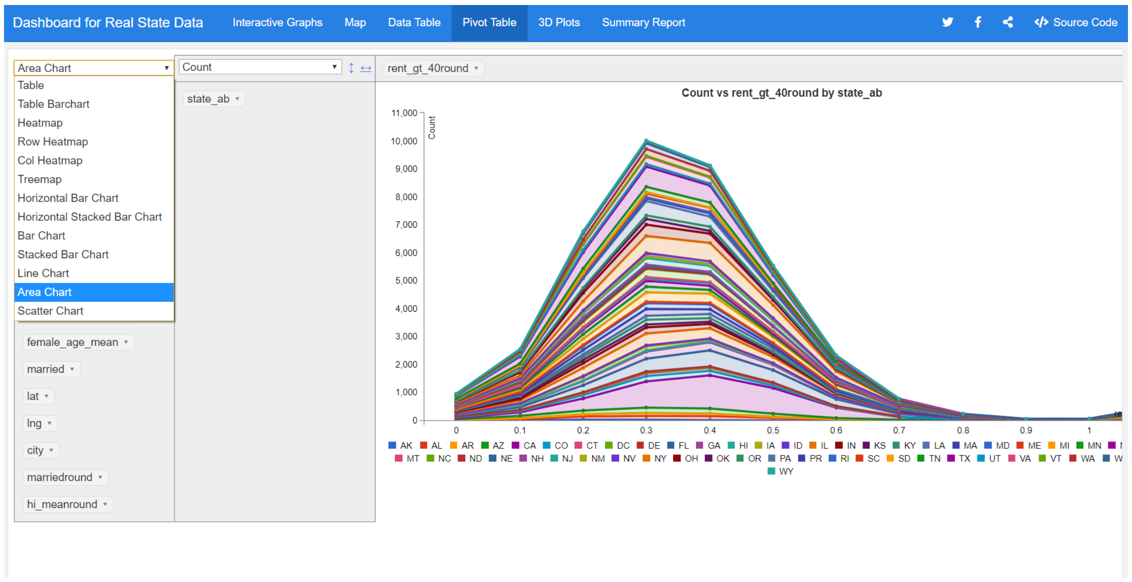


Figure 15: Pivot table – area chart

However, it is worth mentioning that not every type of graphs provided by *rpivotTable* can fit our dataset. Some of them cannot even return a meaningful or applicable graph, but they can be functional in other suitable datasets.

5) 3D plots

3D scatter plot – rent_mean

Plotly library can also create interactive 3D scatter plots or mesh plots. Since we are creating a dashboard, interactive 3D plots are suitable to apply. If we want to draw 3D scatterplots into offline files, *scatterplot3d* function in the **scatterplot3d** package will be a good choice.

To illustrate 3D plots in a transparent way, we select only observations in one state (South Carolina in the first two plots, and Montana in the last two plots) that has relatively low surveyed population so that the points and correlations in plots can be observed clearly.

In the first plot, we will investigate the joint correlation of household income and marriage ratio to average rent in a certain region. First, surveyed regions in South Carolina are screened from the dataset. Rounded average household income is used as x axis, rounded marriage rate is as y axis, and average rent is as z axis. We use *marker* to vary the color for each observation according to average rent. *Colorscale* can return a stepping color pattern in our plot and *showscale* will bring a scale bar on the top right corner in the plot. We use *add_markers* to allow the setting marker to be applied to the plot.

The new argument in this 3D plot is *annotations*. It gives a legend to color scale bar (*Rental Price(US) dollar*), and the position of legend can be adjusted by x and y axes. The background of plot can be changed by set up *plot_bgcolor* and *paper_bgcolor* in the layout.

```
datasub <-  data[data[,"state_ab"] == "SC",]

# to deliver the points clearly,
#we use rounded avg household income and marriage rate
p6 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
                ~rent_mean,
# use marker to select which variable (rent_mean)
# to use as a color criteria;
# colorscale is used to allow distinguished colors
#applied in different levels of rent_mean;
#There are different ready-to-use colorscale in the library,
#One can also customize the colorscale;
              marker = list(color = ~rent_mean,
                            colorscale = 'Viridis',
                            # showscale to allow scale
                            #bar on the top right
                            showscale = TRUE))%>%
   #add_markers to apply the marker we set
   add_markers() %>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                   yaxis = list(title = 'married'),
                       zaxis = list(title = 'rent')),
          #annotation sets up the caption for colorscale bar
          # and its position (x, y)
                      annotations = list(
                        x = 1.16,
                        y = 1.05,
                        text = 'Rental Price(US) dolloar',
                        xref = 'paper',
                        yref = 'paper',
                        showarrow = FALSE
                        ))%>%
   # set up background color of the plot
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p6
```

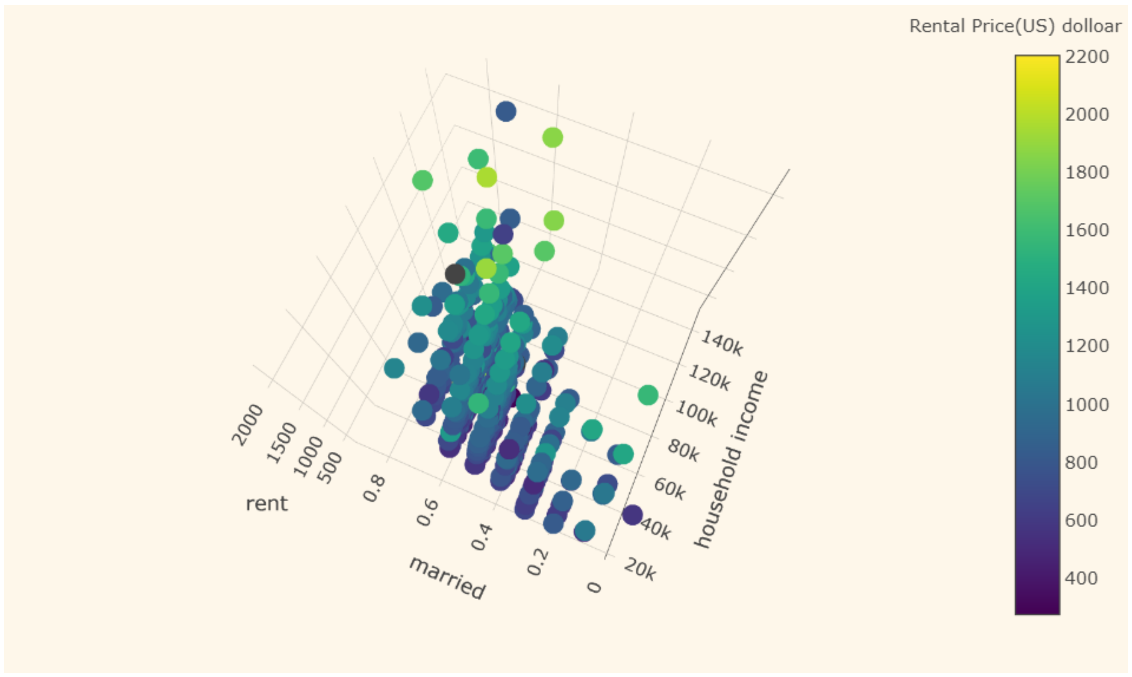


Figure 16: 3D scatter plot – rent mean

It can be seen from the plot that points, with bottom-low rental fees, distribute on those regions with high marriage rate but low household income. Interestingly, those regions with the same level of household income but lower marriage rate, instead, reflect a higher average rental expense. It might be that married people need to allocate more disposable income to food, children schooling, etc. Regional cultures or differing values of generations could also have an effect on it. Therefore, regions with more single people have a tendency to spend more income on rents. On the other hand, regions that can afford the highest rental expenditures are the ones with relatively high household income and high marriage rate.

3D mesh plot – rent_mean

Apart from using scatter plots to represent regional observation, 3D mesh plots are practical to inform us the 3D surface shape of data points. 3D mesh plots depict evident representations for distribution trend, highlighting the peak and bottom points. To call 3D mesh plots, we only need to add *type = 'mesh3d'*. *intensity = ~rent_mean* specify the color scale target in 3D mesh. Color scale can be customized by using *colors = colorRamp*, whose input is a vector of color series.

```
datasub <-  data[data["state_ab"] == "SC",]

# call 3D mesh plot by specifying "type =mesh3d"
# intensity is the criteria for 3d surface color scale
# use colorRamp to define customized color scale
p7 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
                ~rent_mean, type = 'mesh3d',intensity = ~rent_mean,
              colors = colorRamp(c('#002E6E','#266C87','#329691',
                                   '#50AEB5', '#4CBB17')))%>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                   yaxis = list(title = 'married'),
                       zaxis = list(title = 'rent')),
                      annotations = list(
                        x = 1.16,
                        y = 1.05,
                        text = 'Rental Price(US) dolloar',
                        xref = 'paper',
                        yref = 'paper',
                        showarrow = FALSE
                        ))%>%
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p7
```

Our project is faced with a problem when setting up the *colorRamp*. We successfully output 3D mesh plots when they are debugged and run within R chunk. The colors and graphs align with the coding. However, when we knit the 3D mesh plots into flexdashboard/website version, 3D mesh plots, pivot table, data table cannot be shown in flexdashboard. We realize that this problem might come from feeding too many colors into 3D mesh plots. Indeed, when we omit the color customization and use the default basic-blue color, the whole flexdashboard works well again in the website version.

Therefore, in the documents our team hand in contain two different coding for 3D mesh plots, R markdown version for *writing report*, and R markdown for *knitting flexible dashboarding*. 3D mesh coding in report writing version contains more vivid *colorRamp*, while 3D coding in flexible dashboarding will be default color. This might reduce the consistency of 3D plots tab, while it assures the debug.

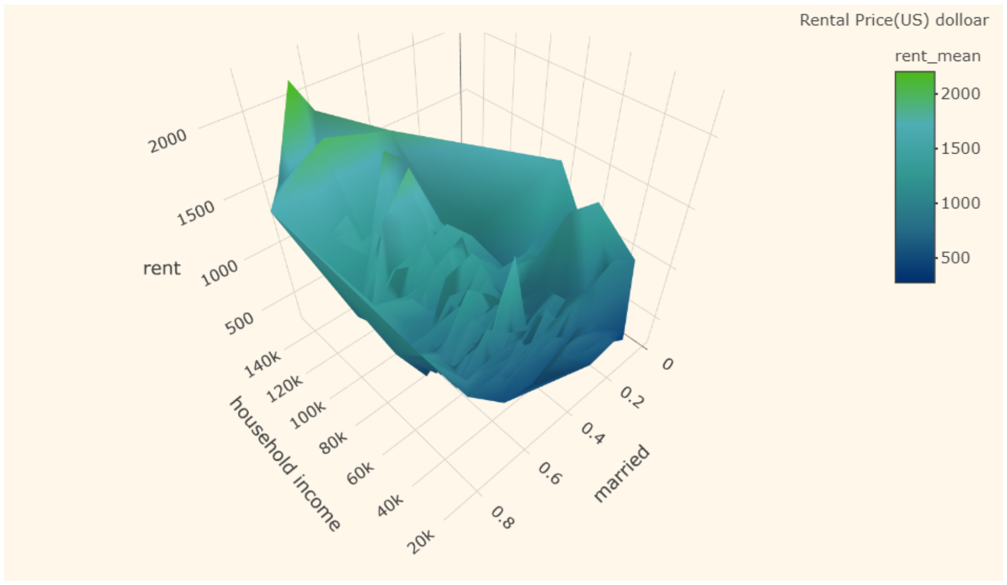


Figure 17: 3D mesh plot – rent mean

3D scatter plot – rent_gt_40round

Similar to the previous 3D scatter plot, this plot is more about financial pressure, percentage of rental expense taking up in house income. *Rent_gt_40 round* measures the percentage population in one region that individual's rent will be greater than 40% of his household income in the past 12 months. It is a good reference for regional financial pressure. The higher criterion represents a more common financial distress in a certain region. We randomly select Montana state to collect surveyed observations.

```r
datasub <-  data[data[,"state_ab"] == "MT",]

p8 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
                ~rent_gt_40round,
              marker = list(color = ~rent_gt_40round,
                            colorscale = 'Portland',
                            showscale = TRUE))%>%
   add_markers() %>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                  yaxis = list(title = 'married'),
                       zaxis = list(title = 'rent perencetage')),
                      annotations = list(
                         x = 1.13,
                        y = 1.05,
                        text = 'rent percentage',
                        xref = 'paper',
                        yref = 'paper',
                        showarrow = FALSE
                        ))%>%
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p8
```

In Montana, two kinds of regions show high *rent_gt_40round*. One is with middle average household income and relatively high marriage rate; another is with both low income and low marriage rate.

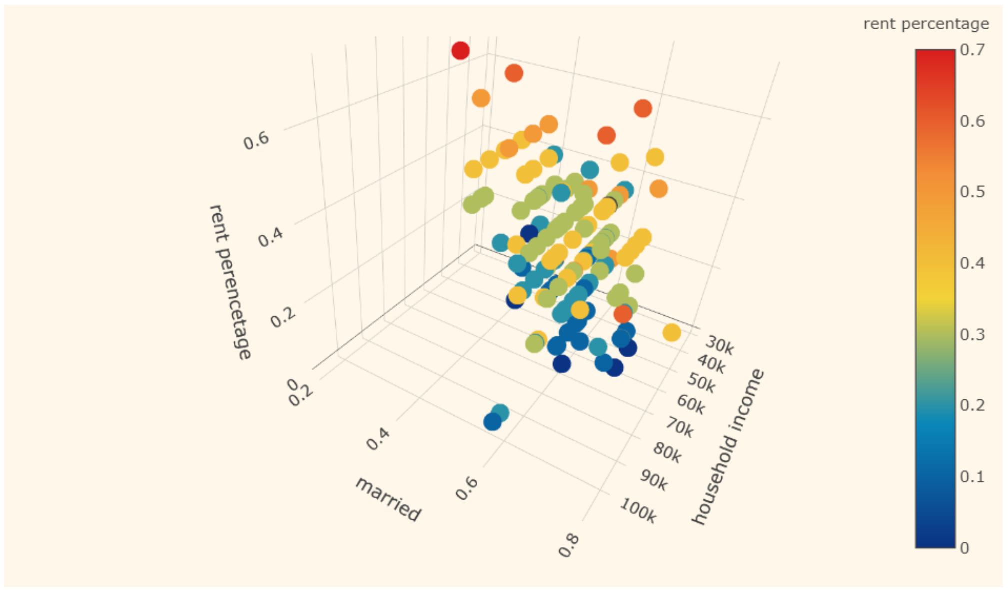


Figure 18: 3D scatter plot – rent gt 40round

3D mesh plot – rent_gt_40round

Mesh plot for *rent_gt_40round* similarly brings the knitting problem. Therefore, we use the same method to treat this mesh plot.

```r
datasub <-  data[data[,"state_ab"] == "MT",]

p9 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
                ~rent_gt_40round, type = 'mesh3d',intensity =~rent_gt_40round,
            colors = colorRamp(c('#4CBB17','#C7EA46','#FFF275','#FF8C42', '#FF3C38',
                                 '#A23E48')))%>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                   yaxis = list(title = 'married'),
                        zaxis = list(title = 'rent perencetage')),
          #calibrate the color scale title
         annotations = list(
                        x = 1.13,
                       y = 1.05,
                       text = 'rent percentage',
                       xref = 'paper',
                       yref = 'paper',
                       showarrow = FALSE
                       ))%>%
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p9
```

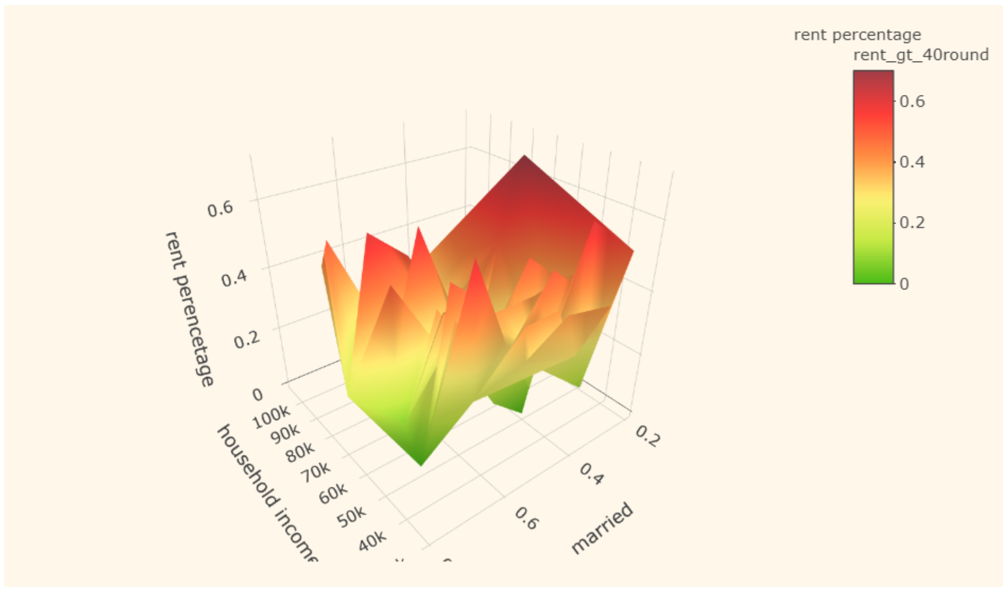


Figure 19: 3D mesh plot – rent gt 40round

6) Summary box

Gauge box

Gauge() function in library **flexdashboard** can help to draw average rental expense gauge. We use the mean of rental expense all surveyed regions as a criterion for American overall housing rent. Minimum and maximum intervals need to be defined for the gauge graph. We can customize different sub intervals as *success, warning and danger parameter*. Then, different colors will be applied to different sub intervals.

```
# impute NA for rent_mean
data$rent_mean <- with(data, impute(data$rent_mean, mean))

# define the minimum and maximum intervasl in the gauge
gauge(round(mean(data$rent_mean),
            digits = 2),
            min = 0,
            max = 2000,
      #define different sections for rental gauge
            gaugeSectors(success = c(0, 700),
                         warning = c(700, 1400),
                         danger = c(1400, 2000),
                         colors = c("green", "yellow", "red")))
```

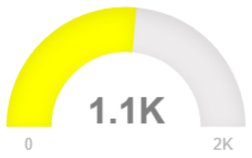


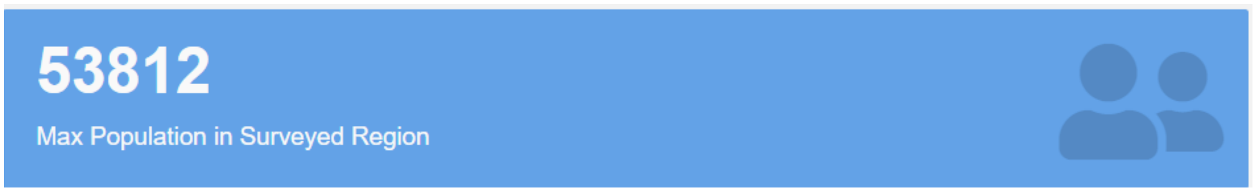
Figure 20: Gauge box

Value box

Value box is a very concise and practical visualization tool in dashboarding. *valueBox* is from library **shiny**. It gives summarizing and comprehensive information for the data set. For example. By using *max(data$pop)*, we can search for the region with the maximum surveyed population. It will be presented with a clear number and concise icon. Value boxes can be applied to present summarizing statistics during daily reports, meeting and presentations.

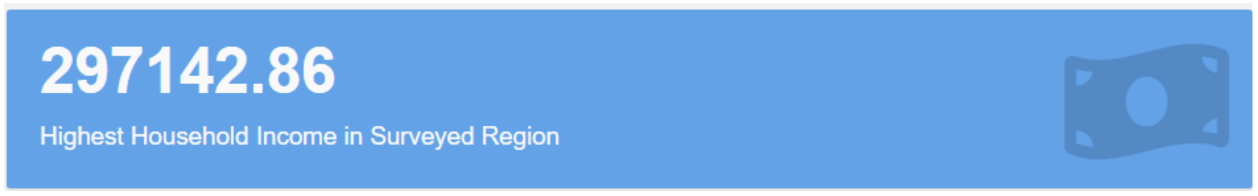
Argument *icon* places various image adaptive to *valueBox* topic. Available lists of *icon* collection can be visited in https://fontawesome.com/icons?from=io

```
valueBox(max(data$pop),
         #Available lists of *icon* collection can be visited
         #: in https://fontawesome.com/icons?from=io
         icon = "fas fa-user-friends" )

valueBox(max(data$hi_meanround),
         icon = "fas fa-money-bill-wave" )

data$rent_mean <- with(data, impute(data$rent_mean, mean))
valueBox(round(max(data$rent_mean),digits=2),
         icon = "fas fa-home" )
```


53812

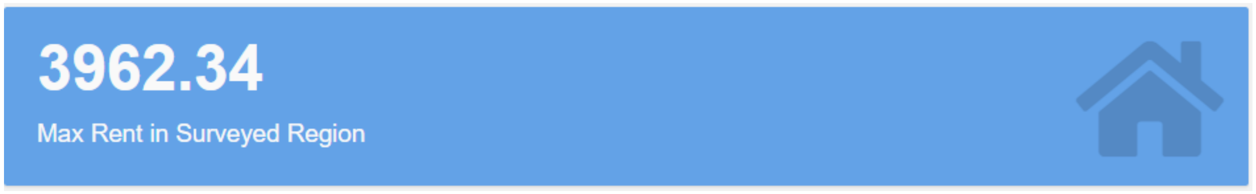
Max Population in Surveyed Region

Figure 21: Value Box 1


297142.86

Highest Household Income in Surveyed Region

Figure 22: Value Box 2


3962.34

Max Rent in Surveyed Region

Figure 23: Value Box 3

6. Reference

Chen, C. H., Härdle, W. K., & Unwin, A. (Eds.). (2007). Handbook of data visualization. Springer Science & Business Media.

Friendly M. (2008) A Brief History of Data Visualization. In: Handbook of Data Visualization. Springer Handbooks Comp.Statistics. Springer, Berlin, Heidelberg

Gorban, A. N., Kégl, B., Wunsch, D. C., & Zinovyev, A. Y. (Eds.). (2008). Principal manifolds for data visualization and dimension reduction (Vol. 58, pp. 96-130). Berlin: Springer.

Kabacoff, R. (2018). Data visualization with R. EEUU: Wesleyan University. Knaflic, C. N. (2015). Storytelling with data: A data visualization guide for business professionals. John Wiley & Sons.

Maniyar, D. M., & Nabney, I. T. (2006, September). Data visualization with simultaneous feature selection. In 2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology (pp. 1-8). IEEE.

Ramachandran, P., & Varoquaux, G. (2011). Mayavi: 3D visualization of scientific data. Computing in Science & Engineering, 13(2), 40-51.

Ward, M. O., Grinstein, G., & Keim, D. (2010). Interactive data visualization: foundations, techniques, and applications. CRC Press.

Wilke, C. O. (2019). Fundamentals of data visualization: a primer on making informative and compelling figures. O'Reilly Media.

Website resources:

https://learn.datacamp.com/skill-tracks/data-visualization-with-r

https://www.data-to-viz.com/

https://www.r-graph-gallery.com

https://www.youtube.com/watch?v=_a4S4tq62OE

7. Appendix

7.1 Website collections for data visualization

R documentation for each function:

package plotly

plot_ly():

https://www.rdocumentation.org/packages/plotly/versions/4.9.2/topics/plot_ly

add_trace():

https://www.rdocumentation.org/packages/plotly/versions/4.9.2/topics/add_trace

package DT

datatable():

https://www.rdocumentation.org/packages/DT/versions/0.13/topics/datatable

package rpivotTable

rpivotTable():

https://www.rdocumentation.org/packages/rpivotTable/versions/0.3.0/topics/rpivotTable

package highcharter

highchart():

https://www.rdocumentation.org/packages/highcharter/versions/0.7.0/topics/highchart

package flexdashboard:

gauge():

https://www.rdocumentation.org/packages/flexdashboard/versions/0.5.1.1/topics/gauge

package shiny:

valueBox():

https://www.rdocumentation.org/packages/shinydashboard/versions/0.7.1/topics/valueBox

For color picking:

https://rgbcolorcode.com/color/air-force-blue-usaf

https://plotly.com/r/colorscales/

For highchart:

highmap collections:

https://code.highcharts.com/mapdata/

geojson:

https://github.com/ropensci/geojson

hc_add_series_map():

https://rdrr.io/cran/highcharter/man/hc_add_series_map.html

function available in highcharter():

http://jkunst.com/highcharter/docs/reference/index.html

charting maps:

https://cran.r-project.org/web/packages/highcharter/vignettes/charting-maps.html

For Shiny:

icon collection:

https://fontawesome.com/

7.2 R code in R markdown

```r
#Interactive Graphs
#======================================
#
#Row
#-------------------------------

### Regions surveyed By State
p1 <- data %>%
         group_by(state_ab) %>%
         summarise(count = n()) %>%
         plot_ly(x = ~state_ab,
                 y = ~count,
# here we don't code it as color=... directly,
#but change it from color to marker=list(color=...)
                 marker = list(color = 'rgb(93,138,168)'),
                 type = 'bar') %>%
layout(xaxis = list(title = "State"),
yaxis = list(title = 'Number of regions surveyed'))
p1


### Top States of participating in the survey
p2 <- data %>%
         group_by(state_ab) %>%
         summarise(count = n()) %>%
   #use filter to filter out states with
   #number of survey regions larger than 1500
         filter(count>1500) %>%
         plot_ly(labels = ~state_ab,
                 values = ~count,
                 marker = list(colors = mycolors)) %>%
   #add_pie belongs to function add_trace(),
   #Add trace(s) to a plotly visualization
   #add_pie to tune more specific parameters related to pie chart,
   #eg. the hole in the center
         add_pie(hole = 0.35) %>%
   #to present only a clear pie chart, we set all lines and grid = F
         layout(xaxis = list(zeroline = F,
                             showline = F,
                             showticklabels = F,
                             showgrid = F),
                yaxis = list(zeroline = F,
                             showline = F,
                             showticklabels=F,
                             showgrid=F))
p2


### Population surveyed in each state
p3 <- plot_ly(data,
              x = ~pop,
              y = ~state_ab,
              marker = list(color = 'rgb(1,106,150)'),
              # use text function to add "name card" for each obs
              text = paste("State:", data$state_ab,
                           "Population:",
                           data$pop),
              type = "bar") %>%
         layout(yaxis = list(title="State"),
                xaxis = list(title = "Population"))
p3


#Row
#------------------------------------

### Scatter Plot of Percentage Married Vs Average household income
data$hi_mean <- with(data, impute(data$hi_mean, mean))
data$married <- with(data, impute(data$married, mean))
data <- data %>%
  mutate(marriedround= round(married, digits = 1),hi_meanround=round(hi_mean,digits=2))

p4 <- plot_ly(data, x=~marriedround) %>%
         add_markers(y = ~hi_meanround,
                     text = ~paste("Average household income: ",
                                   hi_meanround,"married:", marriedround),
                     showlegend = F) %>%
                     add_lines(y = ~fitted(loess(hi_meanround ~
                                                marriedround)),
                   name = "Loess Smoother",
                   color = I("#EFDA97"),
                   showlegend = T,
                   line = list(width=5)) %>%
         layout(xaxis = list(title = "Percentage Married"),
                yaxis = list(title = "Regional average household income"))
p4


### Box Plot of Average household income for each state
```

```r
p4.2 <- plot_ly(data, y = ~hi_meanround, color = ~state_ab, type =
                  "box")%>%
               layout(xaxis = list(title = "State"),
               yaxis = list(title = 'Average houshold income'))
p4.2



#Map
#========================================

### Map

#group by states to sum up number of regions being surveyed
p5 <- data %>%
         group_by(state_ab) %>%
         summarize(total = n())
# use abbr2state to modify the abbreiviation to full name
#of each state in the US
p5$State <- abbr2state(p5$state_ab)

highchart() %>%
   #add general title
         hc_title(text = "Number of regions surveyed in US") %>%
   #add subtitle
         hc_subtitle(text = "Source: USA real estate survey") %>%
# differing map can be chosen from
         hc_add_series_map(usgeojson, p5,
                           name = "State",
                           #value: the name of the columnn in p5.
                           value = "total",

                           joinBy = c("woename", "State")) %>%
   # enable mapnavigation to allow zooming
         hc_mapNavigation(enabled = T)


#Data Table
#========================================
datatable(data,
          caption = "Real State Data",
          # rownames=TRUE, showing the rownames in dataset
          rownames = T,
          # filter can be top or bottom
          filter = "top",
          options = list(pageLength = 25))


#Pivot Table
#=========================================
data <- data%>%
  mutate(rent_gt_40round=round(rent_gt_40,digits=1))
```

```r
rpivotTable(data,
            #string name to define how we like to statistize the heatmap
            aggregatorName = "Count",
            #column
            cols= "rent_gt_40round",
            rows = "state_ab",
            rendererName = "Heatmap")


#Pivot Table
#==========================================
data <- data%>%
  mutate(rent_gt_40round=round(rent_gt_40,digits=1))

rpivotTable(data,
            #string name to define how we like to statistize the heatmap
            aggregatorName = "Count",
            #column
            cols= "rent_gt_40round",
            rows = "state_ab",
            rendererName = "Heatmap")

#3D Plots
#======================================

#Row
#-------------------------------

### 3D Scatter Plot (Household income vs married percentage vs rent mean)
datasub <-  data[data[,"state_ab"] == "SC",]

# to deliver the points clearly,
#we use rounded avg household income and marriage rate
p6 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
                ~rent_mean,
# use marker to select which variable (rent_mean)
# to use as a color criteria;
# colorscale is used to allow distinguished colors
#applied in different levels of rent_mean;
#There are different ready-to-use colorscale in the library,
#One can also customize the colorscale;
              marker = list(color = ~rent_mean,
                            colorscale = 'Viridis',
                            # showscale to allow scale bar
                            #on the top right
                            showscale = TRUE))%>%
   #add_markers to apply the marker we set
   add_markers() %>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                   yaxis = list(title = 'married'),
                        zaxis = list(title = 'rent')),
          #annotation sets up the caption for colorscale bar
          # and its position (x, y)
```

```
                      annotations = list(
                        x = 1.16,
                        y = 1.05,
                        text = 'Rental Price(US) dolloar',
                        xref = 'paper',
                        yref = 'paper',
                        showarrow = FALSE
                        ))%>%
   # set up background color of the plot
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p6



### 3D Mesh Plot (Household income vs married percentage vs rent mean)
datasub <-  data[data["state_ab"] == "SC",]

# call 3D mesh plot by specifying "type =mesh3d"
# intensity is the criteria for 3d surface color scale
# use colorRamp to define customized color scale
p7 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
              ~rent_mean, type = 'mesh3d')%>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                   yaxis = list(title = 'married'),
                       zaxis = list(title = 'rent')),
                      annotations = list(
                        x = 1.00,
                        y = 1.05,
                        text = 'Rental Price(US) dolloar',
                        xref = 'paper',
                        yref = 'paper',
                        showarrow = FALSE
                        ))%>%
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p7


#Row
#-------------------------------

### 3D Scatter Plot (Household income vs married percentage vs percentage of rents greater than %40 of
datasub <-  data[data[,"state_ab"] == "MT",]

p9 <- plot_ly(datasub, x = ~hi_meanround, y = ~marriedround, z =
              ~rent_gt_40round, type = 'mesh3d')%>%
   layout(scene = list(xaxis = list(title = 'household income'),
                                   yaxis = list(title = 'married'),
                       zaxis = list(title = 'rent perencetage')),
           #calibrate the color scale title
          annotations = list(
                        x = 1.00,
                         y = 1.05,
                         text = 'rent percentage',
                         xref = 'paper',
                         yref = 'paper',
                         showarrow = FALSE
                         ))%>%
   layout(plot_bgcolor='rgb(254, 247, 234)') %>%
   layout(paper_bgcolor='rgb(254, 247, 234)')
p9


#Summary Box
#=====================================

#Column
#-------------------------------------

### **Average Rent in US**
# impute NA for rent_mean
data$rent_mean <- with(data, impute(data$rent_mean, mean))

# define the minimum and maximum intervasl in the gauge
gauge(round(mean(data$rent_mean),
            digits = 2),
            min = 0,
            max = 2000,
      #define different sections for rental gauge
            gaugeSectors(success = c(0, 700),
                         warning = c(700, 1400),
                         danger = c(1400, 2000),
                         colors = c("green", "yellow", "red")))


### Max Population in Surveyed Region
valueBox(max(data$pop),
         #Available lists of *icon* collection can be visited
         #: in https://fontawesome.com/icons?from=io
         icon = "fas fa-user-friends" )

#Column
#-------------------------------------

### Highest Household Income in Surveyed Region
valueBox(max(data$hi_meanround),
         icon = "fas fa-money-bill-wave" )

### Max Rent in Surveyed Region
data$rent_mean <- with(data, impute(data$rent_mean, mean))
valueBox(round(max(data$rent_mean),digits=2),
         icon = "fas fa-home" )
```