

To perform Data Visualization using R Language.

What is Data Visualization?

Data visualization is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

The popular data visualization tools that are available are Tableau, Plotly, R, Google Charts, Infogram, and Kibana. The various data visualization platforms have different capabilities, functionality, and use cases. They also require a different skill set. R is a language that is designed for statistical computing, graphical data analysis, and scientific research. It is usually preferred for data visualization as it offers flexibility and minimum required coding through its packages.

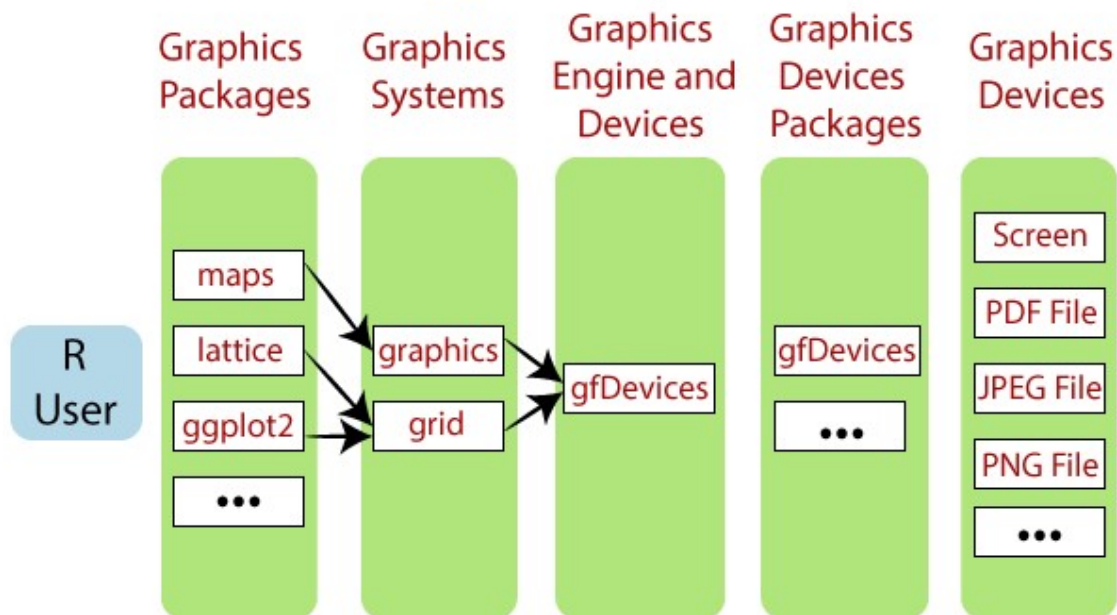
Visualization Packages



R Visualization Packages → R provides a series of packages for data visualization.

1. **plotly**: Provides online interactive graphs, extending the capabilities of plotly.js.
2. **ggplot2**: Known for its elegant and high-quality declarative graphics in R.
3. **tidyquant**: A financial package within tidyverse for importing, analyzing, & visualizing financial data.
4. **taucharts**: Offers a declarative interface for quickly mapping data to visual properties.
5. **ggiraph**: Enables dynamic ggplot graphs with tooltips, JavaScript actions, and animations.
6. **geofacets**: Provides geofaceting functionality for ggplot2, arranging plots for different geographical entities in a grid.
7. **googleVis**: Interfaces R with Google's charts tools to create interactive charts on web pages.
8. **RColorBrewer**: Provides color schemes, particularly for maps & graphics.
9. **dygraphs**: R interface to the dygraphs JavaScript library, specializing in time-series data visualization.
10. **shiny**: Enables creation of interactive web apps with R, leveraging HTML widgets, CSS, and JavaScript extensions.

R Graphics



The basics of the grammar of graphics

There are some key elements of a statistical graphic which are the basics of the grammar of graphics.

- **Data** → Data is the most crucial thing which is processed and generates an output.
- **Aesthetic Mappings** → Aesthetic mappings are one of the most important elements of a statistical graphic. It controls the relation between graphics variables and data variables. In a scatter plot, it also helps to map the temperature variable of a data set into the X variable.
In graphics, it helps to map the species of a plant into the color of dots.
- **Geometric Objects** → Geometric objects are used to express each observation by a point using the aesthetic mappings. It maps two variables in the data set into the x,y variables of the plot.
- **Statistical Transformations** → It allow us to calculate the statistical analysis of the data in the plot. The statistical transformation uses the data and approximates it with the help of a regression line having x,y coordinates, and counts occurrences of certain values.
- **Scales** → It is used to map the data values into values present in coordinate system of graphics device.
- **Coordinate system** → It plays an important role in the plotting of the data.
 - Cartesian
 - Plot
- **Faceting** → Faceting is used to split the data into subgroups and draw sub-graphs for each group.

Types of Data Visualization in R Language

1. Scatter Plot

A scatter plot is composed of many points on a Cartesian plane. Each point denotes the value taken by two parameters and helps us easily identify the relationship between them.

Scatter Plots are used in the following scenarios:

- To show whether an association exists between bivariate data.
- To measure the strength and direction of such a relationship.

2. Box Plot

The statistical summary of the given data is presented graphically using a boxplot. A boxplot depicts information like the minimum and maximum data point, the median value, first and third quartile, and interquartile range.

Box Plots are used for:

- To give a comprehensive statistical description of the data through a visual cue.
- To identify the outlier points that do not lie in the inter-quartile range of data.

3. Bar Plot

There are two types of bar plots- horizontal and vertical which represent data points as horizontal or vertical bars of certain lengths proportional to the value of the data item. They are generally used for continuous and categorical variable plotting. By setting the **horiz** parameter to true and false, we can get horizontal and vertical bar plots respectively.

Bar plots are used for the following scenarios:

- To perform a comparative study between the various data categories in the data set.
- To analyze the change of a variable over time in months or years.

4. Histogram

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

For a histogram, the parameter **xlim** can be used to specify the interval within which all values are to be displayed.

Another parameter **freq** when set to *TRUE* denotes the frequency of the various values in the histogram and when set to *FALSE*, the probability densities are represented on the y-axis such that they are of the histogram adds up to one.

Histograms are used in the following scenarios:

- To verify an equal and symmetric distribution of the data.

- To identify deviations from expected values.

5. Heat Map

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. `heatmap()` function is used to plot heatmap.

Syntax: heatmap(data)

Parameters: data: It represent matrix data, such as values of rows and columns

Return: This function draws a heatmap.

6. Map visualization in R

Here we are using maps package to visualize and display geographical maps using an R programming language.

```
install.packages("maps")
```

Advantages of Data Visualization in R:

- Broad collection of visualization libraries along with extensive online guidance on their usage.
- Customization: Through R, we can easily customize our data visualization by changing axes, fonts, legends, annotations, and labels.
- Understanding: Visual graphics make business insights more appealing and understandable, aiding better decision-making.
- Efficiency: Graphics efficiently organize complex business data, optimizing information management. R also offers data visualization in the form of 3D models and multipanel charts.
- Location: Geographic maps and GIS features visualize location-specific insights, enhancing response strategies.

Disadvantages of Data Visualization in R:

- R application development can be costly for small companies. Hiring professionals to create charts adds expenses, despite the critical need for timely evaluation results.
- Complex data visualizations may prioritize style over substance, risking reduced overall value. It's crucial to use resources wisely and avoid unnecessary graphical trends.
- R is only preferred for data visualization when done on an individual standalone server.
- Data visualization using R is slow for large amounts of data as compared to other counterparts.

Application Areas:

1. Presenting analytical conclusions of the data to the non-analysts departments of your company.
2. Health monitoring devices use data visualization to track any anomaly in blood pressure, cholesterol and others.
3. To discover repeating patterns and trends in consumer and marketing data.
4. Meteorologists use data visualization for assessing prevalent weather changes throughout the world.
5. Real-time maps and geo-positioning systems use visualization for traffic monitoring and estimating travel time.

Methodology:

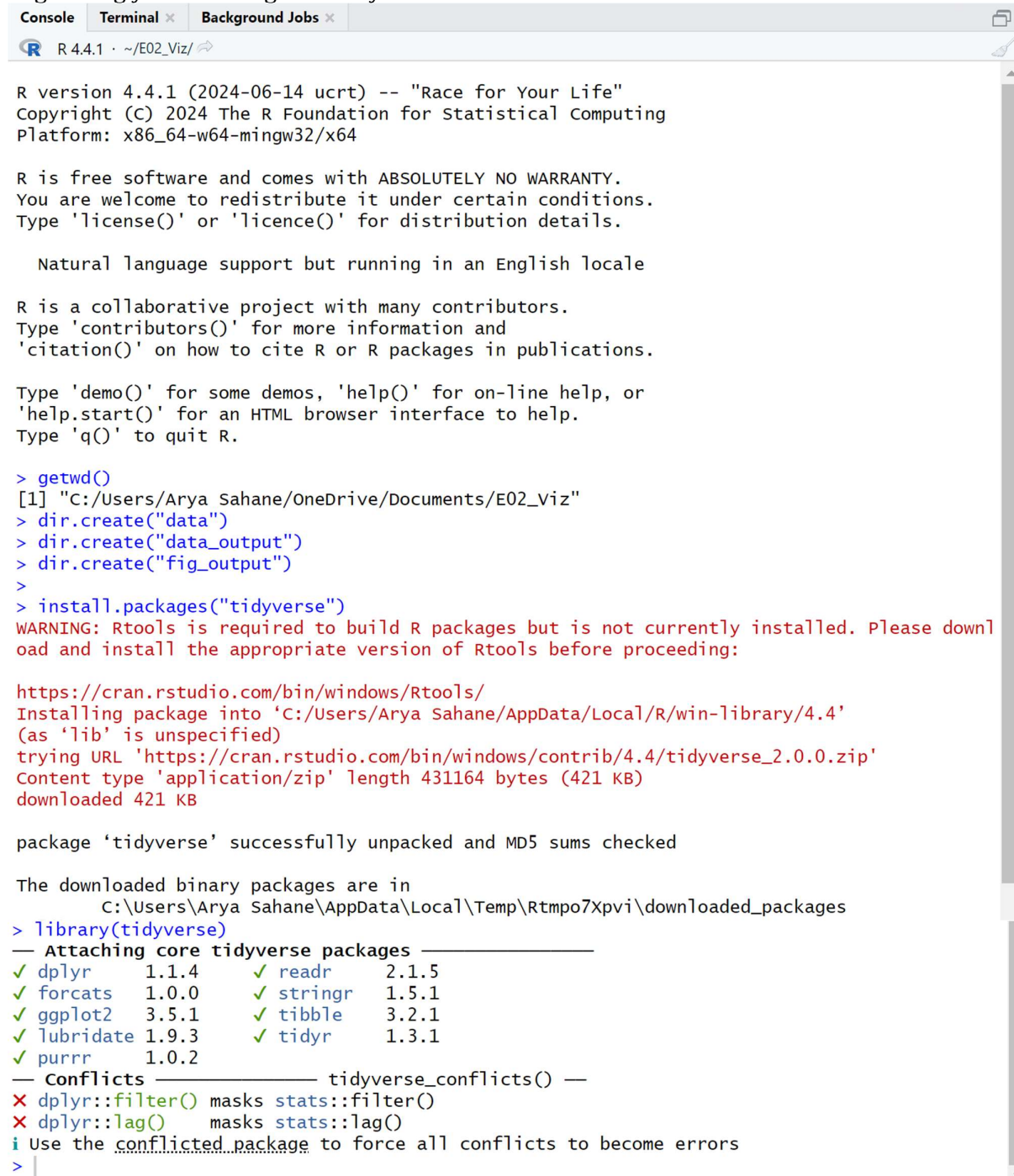
1. **Exploratory Data Analysis (EDA):** Use basic plots like histograms, scatter plots, and box plots to understand data distributions and relationships.
2. **Statistical Graphics:** Utilize statistical plots such as bar plots, pie charts, and density plots to summarize data characteristics.
3. **Advanced Plots:** Employ advanced techniques like heatmaps, contour plots, and treemaps for detailed visual exploration of complex datasets.
4. **Interactive Visualization:** Use packages like `plotly` and `shiny` to create interactive graphs that enhance user engagement and exploration.
5. **Customization:** Customize plots by adjusting axes, colors, legends, annotations, and labels using packages such as `ggplot2` to convey specific insights effectively

Code & Output:

1. Setup R console, libraries & packages required, and Directories →

`getwd()` → ensures that working directory is set up properly.

Organizing your working directory



```
R 4.4.1 · ~/E02_Viz/

R version 4.4.1 (2024-06-14 ucrt) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> getwd()
[1] "C:/Users/Arya Sahane/OneDrive/Documents/E02_Viz"
> dir.create("data")
> dir.create("data_output")
> dir.create("fig_output")
>
> install.packages("tidyverse")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

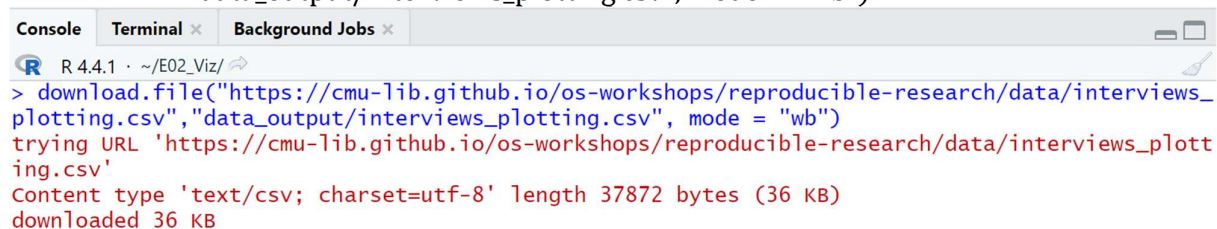
https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/Arya Sahane/AppData/Local/R/win-library/4.4'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/tidyverse_2.0.0.zip'
Content type 'application/zip' length 431164 bytes (421 KB)
downloaded 421 KB

package 'tidyverse' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:/Users/Arya Sahane/AppData/Local/Temp/Rtmpo7Xpvi/downloaded_packages
> library(tidyverse)
— Attaching core tidyverse packages —
✓ dplyr      1.1.4    ✓ readr      2.1.5
✓ forcats    1.0.0    ✓ stringr    1.5.1
✓ ggplot2    3.5.1    ✓ tibble     3.2.1
✓ lubridate  1.9.3    ✓ tidyr      1.3.1
✓ purrr      1.0.2
— Conflicts — tidyverse_conflicts() —
✗ dplyr::filter() masks stats::filter()
✗ dplyr::lag() masks stats::lag()
i Use the conflicted package to force all conflicts to become errors
>
```

2. Download Dataset

```
download.file("https://cmu-lib.github.io/os-workshops/reproducible-research/data/
interviews_plotting.csv",
              "data_output/interviews_plotting.csv", mode = "wb")
```



```
R 4.4.1 · ~/E02_Viz/

> download.file("https://cmu-lib.github.io/os-workshops/reproducible-research/data/interviews_
plotting.csv", "data_output/interviews_plotting.csv", mode = "wb")
trying URL 'https://cmu-lib.github.io/os-workshops/reproducible-research/data/interviews_plott
ing.csv'
Content type 'text/csv; charset=utf-8' length 37872 bytes (36 KB)
downloaded 36 KB
```

3. Store the dataset in a new variable

```
interviews_plotting <- read_csv("data_output/interviews_plotting.csv")
```

R E02_Viz - RStudio Source Editor

interviews_plotting ×

Filter

	key_ID	village	interview_date	no_membrs	years_liv	respondent_wall_type	rooms	memb_assoc	affect_c
1	1	God	2016-11-17	3	4	muddaub	1	NA	NA
2	1	God	2016-11-17	7	9	muddaub	1	yes	once
3	3	God	2016-11-17	10	15	burntbricks	1	NA	NA
4	4	God	2016-11-17	7	6	burntbricks	1	NA	NA
5	5	God	2016-11-17	7	40	burntbricks	1	NA	NA
6	6	God	2016-11-17	3	3	muddaub	1	NA	NA
7	7	God	2016-11-17	6	38	muddaub	1	no	never
8	8	Chirodzo	2016-11-16	12	70	burntbricks	3	yes	never
9	9	Chirodzo	2016-11-16	8	6	burntbricks	1	no	never
10	10	Chirodzo	2016-12-16	12	23	burntbricks	5	no	never
11	11	God	2016-11-21	6	20	sunbricks	1	NA	NA
12	12	God	2016-11-21	7	20	burntbricks	3	yes	never
13	13	God	2016-11-21	6	8	burntbricks	1	no	never
14	14	God	2016-11-21	10	20	burntbricks	3	NA	NA
15	15	God	2016-11-21	5	30	sunbricks	2	yes	once
16	16	God	2016-11-24	6	47	muddaub	1	NA	NA
17	17	God	2016-11-21	8	20	sunbricks	1	NA	NA
18	18	God	2016-11-21	4	20	muddaub	1	NA	NA

4. Prepare a subdataset as percent_wall_type

```
percent_wall_type <- interviews_plotting %>%
  filter(respondent_wall_type != "cement") %>%
  count(village, respondent_wall_type) %>%
  group_by(village) %>%
  mutate(percent = (n / sum(n)) * 100) %>%
  ungroup()
```

R E02_Viz.R* × percent_wall_type ×

Filter

	village	respondent_wall_type	n	percent
1	Chirodzo	burntbricks	22	56.410256
2	Chirodzo	muddaub	16	41.025641
3	Chirodzo	sunbricks	1	2.564103
4	God	burntbricks	19	44.186047
5	God	muddaub	15	34.883721
6	God	sunbricks	9	20.930233
7	Ruaca	burntbricks	26	54.166667
8	Ruaca	muddaub	15	31.250000
9	Ruaca	sunbricks	7	14.583333

NOTE → ggplot2 is an R package that provides a powerful and flexible system for creating and visualizing graphs and plots based on the grammar of graphics.

#ggplot objects SYNTAX

<DATA> %>%

ggplot(aes(<MAPPINGS>)) + # Specify data and aesthetic mappings

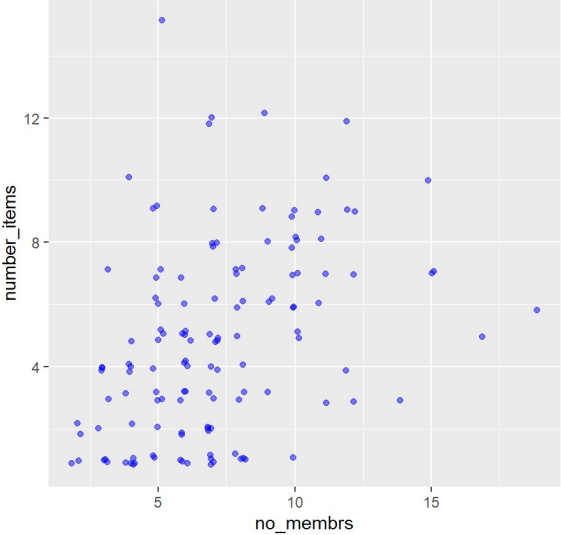
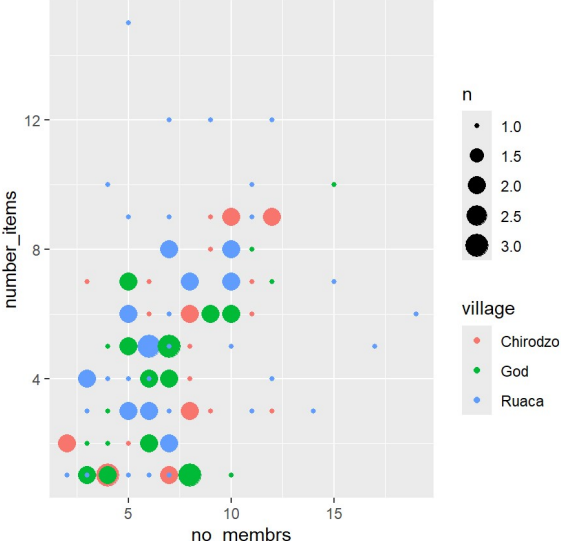
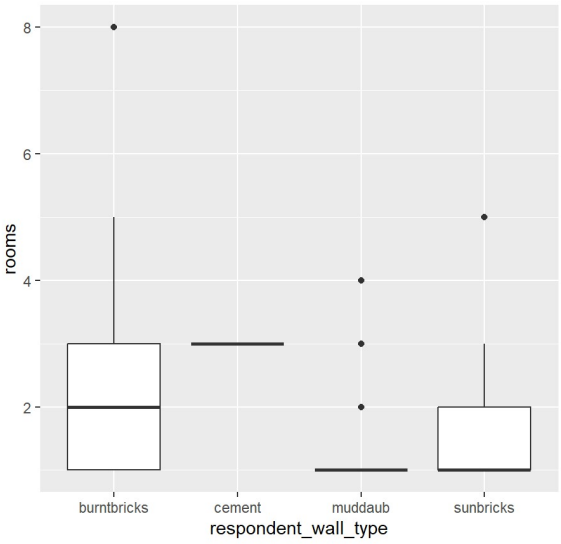
<Geometric_Functions>() # Add geometric layers (e.g., points, lines, bars)

Here's a breakdown of each component:

- <DATA>: This represents the data frame or tibble that contains the data you want to visualize.
- %>%: This is the pipe operator (magrittr pipe), used to pass the data frame into the ggplot() function without explicitly referring to it.
- ggplot(aes(<MAPPINGS>)): The ggplot() function initializes a ggplot object with the data and aesthetic mappings (aes()). <MAPPINGS> typically includes variables from <DATA> mapped to aesthetics like x and y axes, color, shape, etc.
- <Geometric_Functions>(): This part adds geometric layers to the plot, such as points (geom_point()), lines (geom_line()), bars (geom_bar()), etc. These functions specify how the data should be represented visually (e.g., as points, lines, bars) based on the mappings provided in ggplot()

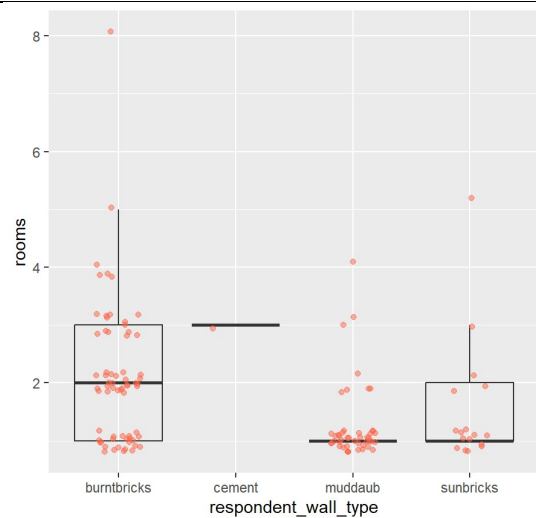
Basic Functions in ggplot2

Name	Syntax	Function
ggplot	ggplot(data, aes(...))	Initializes a ggplot object with the dataset and aesthetic mappings.
geom_point	geom_point(mapping = aes(...), ...)	Adds a scatter plot layer to the plot.
geom_line	geom_line(mapping = aes(...), ...)	Adds a line plot layer to the plot.
geom_jitter	geom_jitter(mapping = aes(...), ...)	Adds a jittered scatter plot layer to reduce overplotting by adding random noise.
geom_smooth	geom_smooth(mapping= aes(...), ...)	Adds a smoothed line layer, such as regression line.
geom_count	geom_count(mapping = aes(...), ...)	Adds a count plot layer where the size of the points represents the count of occurrences.
geom_bar	geom_bar(mapping = aes(...), stat = "identity", ...)	Adds a bar plot layer. Use stat = "identity" if the data is pre-summarized.
geom_boxplot	geom_boxplot(mapping = aes(...), ...)	Adds a boxplot layer to the plot, showing distribution of data based on quartiles.
geom_violin	geom_violin(mapping = aes(...), ...)	Adds a violin plot layer, which combines aspects of boxplots and density plots.
geom_histogram	geom_histogram(mapping = aes(...), ...)	Adds a histogram layer to the plot.
geom_text	geom_text(mapping = aes(...), ...)	Adds text labels to the plot.
geom_label	geom_label(mapping = aes(...), ...)	Adds text labels with a background to plot.
labs	labs(title = "Title", x = "X-axis", y = "Y-axis")	Adds titles and axis labels to the plot.
facet_wrap	facet_wrap(~ variable, ...)	Creates a series of subplots (panels) based on the levels of a variable.
facet_grid	facet_grid(rows ~ columns)	Creates a grid of panels based on two variables.
theme	theme(...)	Customizes non-data aspects of plot, such as background color, text size, & grid lines.
ggsave	ggsave("filename.png", plot, ...)	Saves the current plot to a file.
xlim	xlim(lower, upper)	Sets the limits of the x-axis.
ylim	ylim(lower, upper)	Sets the limits of the y-axis.

CODE	OUTPUT
<p>A) Scatter Plot</p> <pre>#basic scatter plot interviews_plotting %>% ggplot(aes(x = no_membrs, y = number_items)) + geom_point()</pre> <p>2 main ways to alleviate overplotting issues:</p> <ul style="list-style-type: none"> • changing the transparency(opacity: $0 < \alpha < 1$) of the points • jittering(randomness) the location of the points <pre>#scatter plot (jitter & transparency) interviews_plotting %>% ggplot(aes(x = no_membrs, y = number_items)) + geom_jitter(width = 0.2, height = 0.2, alpha = 0.5, color = "blue")</pre>	
<p>B) Point Based Scatter Plot</p> <pre>#point size interviews_plotting %>% ggplot(aes(x = no_membrs, y = number_items, color = village)) + geom_count()</pre>	
<p>C) Box Plot</p> <pre>#boxplot interviews_plotting %>% ggplot(aes(x = respondent_wall_type, y = rooms)) + geom_boxplot()</pre>	

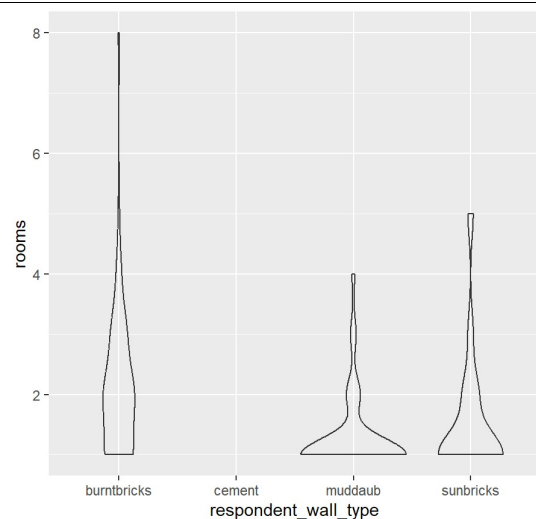
D) Combination of Scatter and Box Plot

```
#Combine plots
interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type, y = rooms)) +
  geom_boxplot(alpha = 0) +
  geom_jitter(alpha = 0.5,
    color = "tomato",
    width = 0.2,
    height = 0.2)
```



E) Violin Plot

```
#violin plots
interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type, y = rooms)) +
  geom_violin(alpha = 0)
```

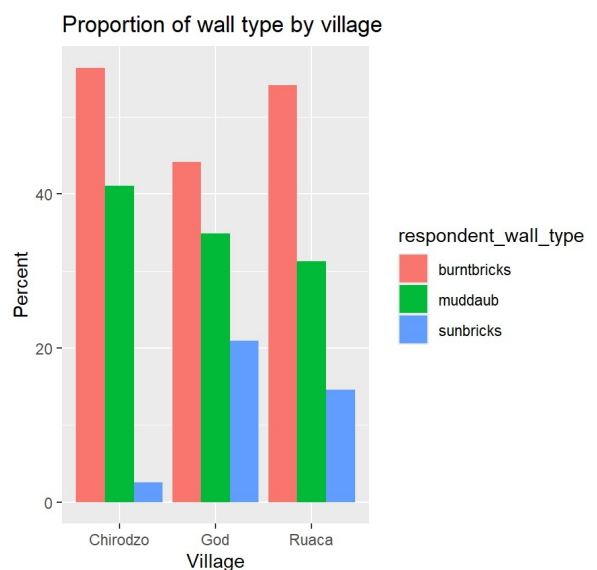


F) Bar Plot

```
#Basic simple barplot
interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type)) +
  geom_bar()
```

```
#aes = village barplot divided column
interviews_plotting %>%
  ggplot(aes(x = respondent_wall_type)) +
  geom_bar(aes(fill = village))
```

```
#Labels
percent_wall_type %>%
  ggplot(aes(x = village, y = percent, fill =
    respondent_wall_type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Proportion of wall type by village",
    x = "Village",
    y = "Percent")
```



<https://cmu-lib.github.io/os-workshops/reproducible-research/Data%20Visualization%20with%20R.pdf>

<https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>