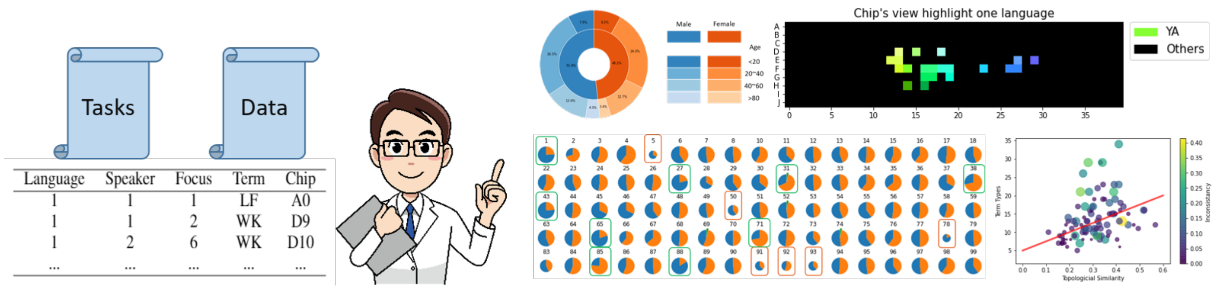
Visualizing World Color Survey Dataset

Yi (Joshua) Ren



Language	Speaker	Focus	Term	Chip
1	1	1	LF	A0
1	1	2	WK	D9
1	2	6	WK	D10
...	...	...	...	...

Abstract—World Color Survey (WCS) project is widely used in many research fields, e.g., cognitive science and evolutionary linguistics. Researchers in these fields believe this dataset can provide useful insight and supportive evidence to their hypothesis. However, as the data is stored in text form, it is hard for the fieldworkers to directly get intuitions from the raw dataset. Data visualization is indeed a powerful tool that can help them know the dataset better, but to the best of our knowledge, there is no open-source code for visualizing the WCS dataset. Hence in this paper, we propose a bunch of visualization designs based on the discussions with the experts in related fields. We conclude five representative tasks and many corresponding solutions and examples. The visualization designs and software implementation we proposed can act as a scaffold during their whole project.

Index Terms—WCS dataset, visualization, distribution, quantitative metrics

1 Introduction

Language is usually considered as the most amazing ability that can tell we human apart from other animals: we use language to communicate, cooperate, express out motions, and transfer knowledge to our descendants. Different from other communication strategies applied by animals, human language usually exhibits striking structures [3], e.g., compositionality allow us to represent complex concept using the combination of several simpler concepts; consistency allows us to remember a group of concepts at the same time. Inspired by these facts, many linguists and cognitive scientists worked for decades to uncover the mystery behind the origin and evolution of human language. As the space of language is too large to handle, some of them narrow down their focus to a small group of concepts, e.g., common objects, numerals, etc.

Among these concepts, one of the widely studied topics is the color semantic domain. The researchers speculate that the language used by participants from different regions (or time) can provide useful hints on the language evolution process. Hence they conduct a project called Word Color Survey (WCS for short) to collect the language data from some settlements around the world [13]. Based on this project, many hypotheses are supported and many insightful works are published – the research field receives a significant breakthrough based on this project.

However, as the data are stored in text form in the WCS project, it is hard for the users to directly get inspiration. As claimed by the authors in their textbook Visualization, Analysis and Design (VAD) [10], an appropriate visualization tool for the data can act as a scaffold for the researchers who want to find some trends from the raw data. After interviewing with some fieldworkers in this direction, we achieve a consensus that a flexible data visualization tool for WCS would be helpful for the researchers in this field. Although many of the related works, e.g., [3, 5, 7, 12, 14], provide some visualization designs for this dataset, there are still many limitations of them, which makes our work necessary. The first one is that the authors of these works generate figures to demonstrate their hypothesis. In other words, the generated figures are not purely objective: the authors might apply reasonable pre-processing or assumptions (as they claimed in their paper) to change the dataset, which is not suitable for the researchers who what to see the visualization of the raw dataset. Second, after carefully searching on the internet, we rarely find the open-source code for generating these figures – the researchers must write code themselves even if they only want to have a glimpse at this dataset.

• *renyi.joshua@gmail.com*

Combining the research requirements in evolutionary linguistics and principles discussed in VAD, we propose a series of visualization designs (so as a software toolkit) to assist a broad range of downstream tasks. The requirements and tasks are concluded from the discussion with fieldworkers. Specifically, we focus on the following three aspects of the dataset in this paper:

- Data visualization for the demographic distribution of the participants;
- Data visualization for the naming of single language;
- Data visualization for the calculated quantitative metrics.

The motivations, detailed designs, and some application examples of them are provided in the rest of the paper. Generally speaking, our paper aims to provide researchers a toolkit to help them generate appropriate visualizations during their whole project: from the brainstorming phase to writing reports. In section 2, we will briefly introduce the WCS dataset and five tasks we plan to solve in the paper. In section 3, we will provide a detailed analysis and design for the visualization of demographic information of participants, with which, the researchers can easily rule out the languages that do not satisfy their requirements. In section 4, we provide the designs for visualizing a single language from two perspectives, with which the researchers can get an intuition of the pattern of that language. In section 5, we provide some examples of observing the correlations among multiple quantitative metrics. In section 6, we make some discussions about the implementation of our software toolkit, so as some limitations and our future work. Finally, we conclude the paper in section 7.

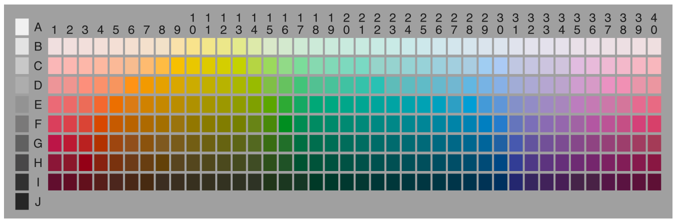


Fig. 1. Standard Munsell color card.

Table 1. Overview of the files in the dataset.

File name	Description
inst.txt	Instructions to fieldworkers
chip.txt	Mapping from chip ID to chip coordinate
foci.txt	Results
foci-exp.txt	Same with foci.txt, but in a different form
spkr.txt	Demographic information of participants
dict.txt	Mapping from terms to abbreviations

2 Data and Task Abstractions

In this section, we first provide a general overview of the WCS project: what is the goal of the project, who are the participants of it, what task they are asked to do, and what kind of data do they generate. Then we will briefly introduce the tasks we plan to solve. After interviewing with some researchers in evolutionary linguistics and cognitive science, we find it would be quite helpful to provide some VAD principles and software tools to assist them. As the raw WCS dataset is stored in the form of text, which is hard for researchers to get intuitions, the data visualizations we proposed can scaffold their future research. The detailed motivation for each task they concern about, so as their requirements and our solutions, are discussed in the rest of this paper.

2.1 Overview of WCS Project and Dataset

To study the language evolutionary process, many researchers narrow down their focus from the entire language (which is used to explain all possible concepts in the world) to some simple but representative tasks. One of the well known task is the world color survey (WCS) project [13], in which volunteers from different regions are asked to describe different colors using their first languages.

Specifically, they use a standard Munsell color card in this task, as illustrated in Figure 1. In this color card, 330 different colors are arranged in two groups: the left column contains blocks with 10 different grayscales (brightness, denoted by $\mathbb{B} = \{A, ..., J\}$) and the right region contains 320 different colors. In the right region, there are 40 columns labeled as $\mathbb{H} = \{1, ..., 40\}$ representing different hues, and 8 rows with $\mathbb{B}_{color} = \{B, ..., I\}$ representing different brightness. Note that in the right region, the brightness only ranges from B to I, because A is pure white and J is pure black. All of these colors are **randomly** indexed with a chip number ranges from 1 to 330. Hence we can refer to a specific color whether using its index (denoted by $idx \in \{1, 330\}$) or by a tuple $(hb) \in \mathbb{H} \times \mathbb{B}$, e.g., $(A01)$. The indexing of the color card can be found in the *chip.txt* file in the WCS dataset. Following the instructions, the participants will name some randomly selected chips using their first language and store the results in *foci-exp.txt*.

To provide a better overview of how information is stored in this dataset, we give Table 1 as a reference. Note that we only list the files required in this paper, the reader can refer to `http://www1.icsi.berkeley.edu/wcs/data.html` for more information.

2.2 Data and Tasks for Visualizing Demographics of Participants

In the WCS project, the volunteers are selected from 110 different regions, i.e., the dataset has 110 different language types. There are on average 24 native speakers for each language. To get knowledge from the ancient languages, some of the participants are carefully selected from those pre-industrialized cultures that had limited contact with modern, industrialized society. The gender and the age of all the participants are stored in *spkr.txt*. Note that even though the questionnaire of the WCS project only provides two options for the gender (i.e., male and female), there are still many recordings of a question mark or blank in this column. Hence we treat these recordings as ‘others’, indicating the participants not willing to tell their gender.

From the discussion with the researchers in evolutionary linguistics and machine learning, we find that a balanced sampling of participants is the prerequisite for an unbiased conclusion. For example, in [6], the authors claim that if the distribution of the gender of samples is not uniform, the derived results will also be biased. Thus before diving into the color-naming task, we need to visualize the demographic information of all the participants involved. In this paper, we mainly concern with the distribution of gender and age.

Domain-specific data:

The domain-specific data can be described in a four-tuple, i.e., (Language ID, Participant ID, Age, Gender), which is also the storage form of items in *spkr.txt*.

Abstracted data:

In each item, the language ID and participant ID are natural numbers that can be considered as unordered classes. The age is an integer from 5 to 100, which can be divided into several intervals when visualizing the distribution. The gender can be considered as an unordered class variable with three possible values (male, female, and not to tell).

To visualize the distributions, we will first clean the data and make sure that all the remaining items are valid, e.g., age is between 5 to 100. After that, we can make some statistics about the data depending on what information we plan to see in the figure. Based on the discussion with the researchers in target fields, we might have the following tasks:

Task 1.1: accumulated distribution under specific conditions

Sometimes, the researchers might want to see the distribution of the gender (or the age) of specific groups of participants. For example, they may want to see the ratio of the male to the female of a group of participants of different ages. With the help of this information, they can decide whether to include the recordings of a specific age range in their analysis.

Task 1.2: detailed distribution for different languages

Sometimes, the researchers are curious about the distribution of all the languages. They may want to compare the total amount of participants and the male-to-female ratio at the same time. With the help of this information, they can choose the languages that satisfy their requirement.

2.3 Data and Tasks for Visualizing a Single Language

The goal of the WCS project is to provide insights of the language evolution process. Usually, the researchers will first select some representative languages from the entire dataset (possibly, with the help of the information provided by Task 1.1 and 1.2), and then draw conclusions from the naming results.

Regarding the naming results, each participant will first read the instructions, and then name different colors on the Munsell using their mother language. The instructions require the participants use one phrase with two words or one word with two syllables to describe each chip in Munsell, which can be considered as an implicit hint that encourages participants to describe two attributes (i.e., hue and lightness) of that chip. As these languages are using different alphabets, the researchers use abbreviations (one or two capital letters) for each phrase. For example, in language 1, *gbanagbana* is mapped to *GB* in the recordings. The mappings are stored in *dict.txt*.

Domain-specific data:

The domain-specific data can be described in a five-tuple, which has the same form of the dataset illustrated in Table 2. There are five columns in this table:

- Language ID: denote which language this piece of recording belongs to, ranging from 1 to 110;
- Speaker ID: denote which speaker this piece of recording comes from;

- Focus response: sequential enumeration of focus responses, not important here;
- Term used: the abbreviation of the term used to describe the chip;
- Described chip: the (bh) representation of the chip described in this recording.

Table 2. Form of the data items in WCS dataset.

Language	Speaker	Focus	Term	Chip
1	1	1	LF	A0
1	1	2	WK	D9
1	2	6	WK	D10
...	...	...	...	...

Abstract data:

In each recording, the 'language ID' and 'speaker ID' are both non-ordered class. The 'term used' attribute is a string composed of 1 or 2 capital letters, which can also be considered as a non-ordered class. The 'described chip' is the coordinate of the chip in the Munsell color card.

From the interview with the researchers in this field, one of the most important concerns of them is the pattern of a single language: whether there exists a pattern, what kind of pattern, or even how the pattern evolves. Under such requirements, we discuss the following tasks as examples:

Task 2.1: patterns of a single language in the grid of Munsell card

Sometimes, the researchers want to see whether the terms used by different languages have similar clusters on the grid of the Munsell card. They speculate that the adjacent chips in the Munsell card might share the same or similar terms, i.e., the results might have clusters. They might want to observe the different clusters of distinct terms in a single language.

Among all the possible patterns, researchers feel most interested in the patterns that can exhibit compositionality. That is because, in evolutionary linguistics, compositionality is usually considered the most important feature of human language. With the help of compositionality, we can easily express a complex concept using the combination of some simple concepts [8]. For example, in English, we might call the chips $B30$, $C30$, and $D30$ 'light blue'. Similarly, we call the chips $B01$, $C01$ and $D01$ 'light red', and the chips $G01$, $H01$ and $I01$ 'dark red'. Obviously, English has compositionality in this WCS task, because in these phrases, the first position represents the brightness and the second position represents the hue. Furthermore, the word 'light' always refers to a color with high brightness (i.e., the first coordinate of the chips are B, C or D), and the word 'red' always refer to color with the same hue (i.e., the second coordinate of the chip is 01). The researchers believe that such compositionality should also exist in the languages collected in the WCS project. They might want to see directly from the data visualization that whether a language is highly compositional.

Task 2.2: the expressivity of specific term(s)

In linguistics, different words might have different expressivity power (e.g., there is plenty of polysemants[1] in many languages) [4]. The researchers might want to see whether the adjectives for colors have distinct expressivity power. For example, in Chinese, the term 'Bai' can represent pure white, light yellow, and light blue (but not light red or light green). Even though there is not a consensus on how such a distinction originates (some may claim that such a distinction might originate from a human's cognitive and vision system), they do need some evidence on the existence of such a distinction. So they wish to see the expressivity of specific terms in the WCS dataset.

[1]The word with multiple distinct meanings. For example, in English, the word 'bank' can represent a financial institution or a slope land beside the river.

2.4 Data and Tasks for Visualizing Quantitative Measurements

For many works based on WCS dataset, researchers will calculate some quantitative measurements for different attributes from the raw data of WCS and observe the influence of these metrics. For example, [1] proposes a metric called TRE to measure the structures of one language, [14] analyzes the mutual information between chips and terms, [2] applies topological similarity to measure the compositionality of a language. What is the correlation among these quantitative measurements then becomes an interesting topic to explore. As this work focus on visualization design rather than evolutionary linguistics, we will not go deeper into these metrics. Rather, we will introduce some vis idioms based on three fundamental and widely applied metrics to show how our visualization toolkit can help.

Domain-specific data:

We need to calculate some quantitative metrics from the raw dataset, the metrics we discussed in this paper are:

- Topological similarity (topsim for short), proposed in [2], a widely used metric for the compositionality of one language;
- Inconsistency (incons for short), a metric to evaluate the extent of divergency of mappings between terms and chips;
- Term types, a metric to evaluate how many different terms are applied in describing all chips on the Munsell card.

Abstract data:

For the aforementioned three metrics, topological similarity is a real number ranging from 0 to 1. Inconsistency is a ratio metric (number of diverged mappings to the total mappings) ranging from 0 to 1. Term types is a natural number.

Based on the discussion with the fieldworkers, we might propose the following task:

Task 3.1: correlation among multiple metrics

In many studies, the linguists claim that the total amount of the commonly used word types should be negatively correlated to the structures of that language [3], but others claim an opposite trend. So researchers might curious about whether such a correlation still exists in the WCS dataset. Furthermore, they might also want to see the correlation between multiple quantitative metrics, e.g., topological similarity, inconsistency, and term types.

2.5 Summary of Data and Task Abstractions

In this section, we briefly introduce the structure and content of the WCS dataset. After that, we discuss several tasks the fieldworkers might care about. The domain-specific data and abstract data are also mentioned for each group of tasks. We also provide a working flow analysis for this paper in Table 3. In the following sections, we will propose our visualization solutions to all the tasks mentioned above. Following the principles provided in [10], we will justify our design by analyzing and comparison with alternative possibilities.

3 Solutions for Visualizing Demographics of Participants

As mentioned in the previous section, sampling the participants from a reasonable distribution is the prerequisite for future research. Hence the first thing the fieldworkers might want to observe is the demographic information of the participants. To better understand their requirements, we conduct a what-why-how analysis, as shown in Table 4.

3.1 Accumulated Distribution under Specific Conditions

For Task 1.1, we may conclude that the researchers care more about whether the male-to-female ratio is unbiased under specific conditions. The conditions might be different languages or different age ranges. To highlight the male-to-female ratio, we may refer to the principles provided in Figure 7.19 in [10], which claims that the pie chart is suitable for demonstrating a part-whole relationship. To further illustrate the information of participants in different age ranges, we propose a

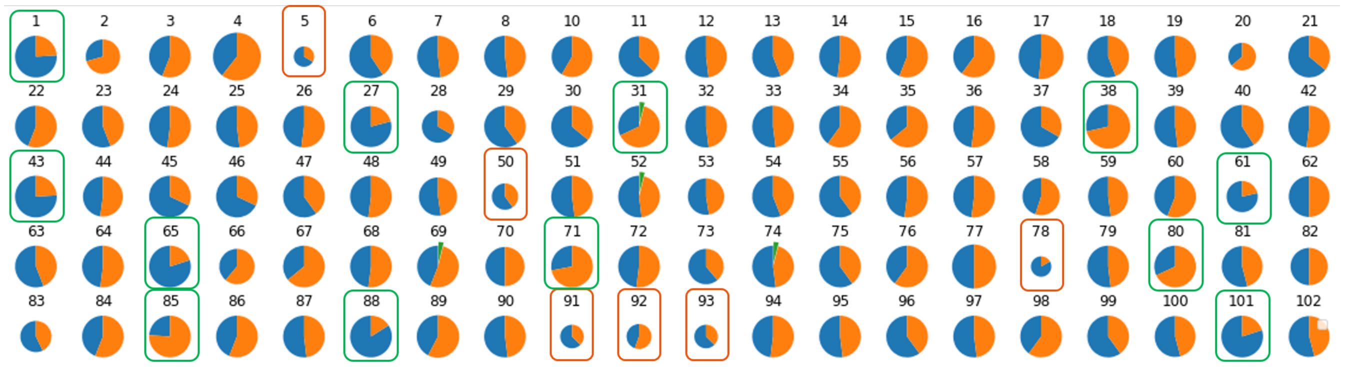


Fig. 2. Example of pie charts for all languages. The number on top of each chart is the language ID. The blue region represents the male while the red region represents the female. The green region (only for some languages) represent the 'Not to tell' participants. On top of the figure, the red square representing a language with too few participants while the green square representing a language with a bias male-to-female ratio.

Table 3. Working flow analysis of the paper. The detailed design are provided in the next section.

Raw Data	'spkr.txt' for the demographic information (age, gender) of participants. There are 110 languages, each with on average 24 participants. 'foci-exp.txt' for the responses from each participant. Each participant will name roughly 20-50 random chips on Munsell card.
Derived Data	The statistics on the demographic data. Three quantitative metrics.
Task	Help the researchers in the target field during their whole project: from the brainstorming phase to report writing. Provide them intuitions.
Interview	Interview with the field workers and narrow down their requirements to different tasks.
Design	Based on the requirements of each task, provide visualization designs following the principles provided in the textbook and related papers.
Implement	Choose a software package and implement the visualization design.
Improve	Improve the design with users and tutor.

Table 4. What-why-how analysis for Task 1.1 and 1.2.

What	Gender (Male, Female, Not to tell) and age (5 to 100) of all participants.
Why	Make sure the accumulated male-to-female ratio is unbiased for languages we focus on. Make sure the age unbalance in terms of gender ratio. Rule out the languages with too few participants (too low statistical significance). Rule out languages with too biased male-to-female ratio.
How	Single chart for Task 1.1. Multiple aligned charts for Task 1.2. Pie-chart to highlight biased male-to-female ratio. Bar-chart to highlight language with too few participants. Pie-ring design to combine gender and age information. Order and align charts using different strategies. Red and blue color to encode male or female. Colors with different alpha value to encode age range.

design called pie-ring chart in Figure 3, from which we can clearly see the nuance differences in the male-to-female ratio, so as the differences among different age ranges. In this pie-ring chart, the inner ring represents the total ratio of male (or female) participants, while the outer ring shows the detailed ratio of participants of different ages. We use different hues (blue and red) to represent gender and different transparency (alpha value) to represent age. The specific percentage is also shown in the corresponding region. From the figure, we can have many interesting conclusions, e.g., we have slightly more male participants; most of the participants' ages are between 20 to 40; the participants with age less than 20 and are slightly more biased to the female while the participants with age from 20 to 60 are slightly more biased to the male.

The shortcoming of this design is that we cannot see the details of each language in the same figure: the design is only suitable for precise comparisons of the accumulated results for a group of languages (the user can select which languages they want to see). It is possible that the accumulated male-to-female distribution is unbiased, but the ratio is biased in some individual languages. To counter this problem, we need Task 1.2.

3.2 Detailed Distribution for Each Language

Different from Task 1.1, the researchers want to obtain a panoramic view of the distribution over all possible languages rather than focusing on a small group (or accumulated results) of languages. They may use this information to rule out some languages before they continue their analysis. Under this requirement, we should first design a meta graph for each language, and then align them appropriately. For the design of meta graph, the pie chart and stacked bar chart has their pros and cons. As mentioned in Chapter 7 in [10], it is easier to tell whether the numbers of male and female participants are the same in pie chart because it is easier for the user to compare two values when they are aligned together than concatenated together. On the other hand, the stacked bar chart is capable of accurately encoding the entire amount of participants, which is hard to obtain in the pie chart. Hence in for Task 1.2, we propose two designs.

The first design is a bunch of aligned small pie charts, as illustrated in Figure 2. From this figure, it is obvious that in some languages, the male-to-female ratio is quite biased, as highlighted by the green square on top of that language. So even though we have an unbiased accumulated male-to-female ratio for all the languages, as illustrated in Figure 3, it is safer to rule out those biased languages before further analysis. Note that we also encode the total number of participants using the size (area) of each pie chart. Via this information, we can also rule out those languages with too few participants, as highlighted by the red square. Remember that the pie chart is good at showing nuance differences in the ratio (using angle) rather than the total amount. Hence we also propose an alternative design based on a stacked bar chart for this task.

As we still want to demonstrate the total amount of participants for each language, we choose a vis idiom similar to the one proposed in

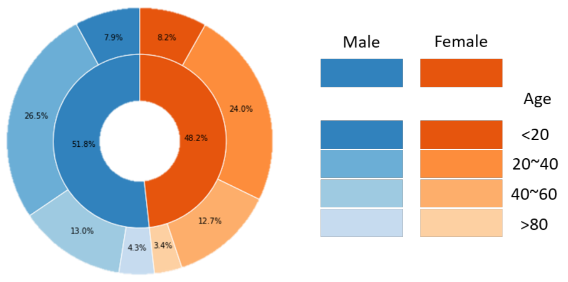


Fig. 3. Example of illustrating the accumulated distribution (of participants in all 110 languages) of male-to-female and the different age range using a pie-ring design.

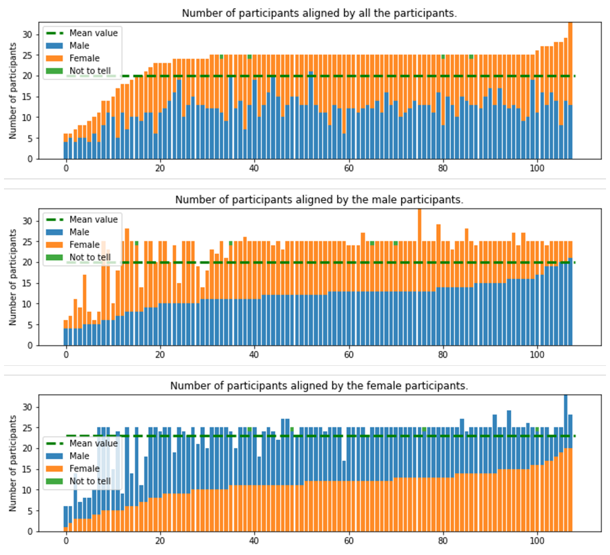


Fig. 4. Example of illustrating the detailed distribution (of participants in all 110 languages) of male-to-female using a stacked bar design. In these three examples, we align the bars by different principles. One shortcoming of these reordered stacked bar chart is that we cannot directly tell the language ID from the figure: we must refer to the code for generating them.

Figure 7.18 in [10]. As shown in Figure 4, the x-axis is the ID of different languages and the y-axis represents the number of participants. Each bar in this chart has two parts: the blue-colored part of male participants and the red-colored part of female participants. The entire length of each stacked bar can represent the total amount of participants speaking this language. To get a better overview of the whole dataset, one can also select how to order the stacked bars, e.g., order them by the number of male participants, by the number of female participants, by the ratio of male/female participants, by the total amount of participants, or by the language ID. With the help of this stacked bar chart, the user can rule out the language with too few participants or select the languages with unbiased male-to-female distributions. The shortcoming of this design is just the strength of the design in Figure 2, i.e., it is hard for the users to notice the nuance bias in the male-to-female ratio.

3.3 Discussions of Demographic Design Choices

The tasks discussed in this part only focus on the distribution of the number of different participant types, hence the pie chart and the stacked bar chart are enough. Also, these two vis idioms have their pros and cons, the users can refer to the following principles:

Pie chart is suitable when:

- you want to show the ratio of aggregate statistics;
- you want to convince the audience that one part occupies a larger ratio than another when their difference is nuance;
- you only want to put a few classes in the figure.

Stacked bar chart is suitable when:

- you want to see distribution of many classes (language or age) simultaneously;
- you want to order these classes to help you to select the satisfying class(es).

4 Solutions for Visualizing a Single Language

Compared with the previous section, the data visualization design of this section focuses more on the language itself. However, there are still nuance differences between Task 2.1 and 2.2. In Task 2.1, the researchers care more about the structure of a specific language, hence they might want to directly observe the pattern of that language, which leads us to consider more from the chip's perspective. On the other hand, Task 2.2 requires us to consider more about the terms in the language, i.e., we should provide a term's perspective to demonstrate their expressivity. Hence in this section, based on the perspective we chose, i.e., the chip's view or the term's view, we might have two different designs. We also provide a what-why-how analysis for the tasks in this part in Table 5.

Table 5. What-why-how analysis for Task 2.1 and 2.2.

What	Responses from each participant, in the form of (language ID, participant ID, Term, Chip ID).
Why	Find patterns, e.g. compositionality, in one language. Find expressivity of each term in one language.
How	Chip's view: Use Munsell grid as the canvas. Color encoding: Munsell color/hue/lightness. Class reduction: cut-off or group merging. For unrecorded chips: interpolation. Term's view: Use stacked-bar fashion design. Color encoding: Munsell color. Length encoding: same length for each chip. Layout: order by length or by term.

4.1 Design from Chip's View

One of the key words in Task 2.1 is **pattern**, which guide us to consider the spatial location of the chips on the Munsell color card. As mentioned in [12], the semantic space of colors in the Munsell card should be continuous and should follow the adjacent principle, i.e., the adjacent chips tend to have the same or similar terms. Hence for the design from the chip's perspective, we use the grid of the Munsell card as our coordinate system, which matches our intuition well. Actually, we can also consider the location of each chip on the Munsell card as geometric information, as discussed in Chapter 8.3 in [10]. Hence we can use a design similar to the Choropleth Map (Figure 8.2 in [10]): we use the Munsell grid as the spatial layout and use color mapping to encode different classes (i.e., terms in this task). Note that the chips represented by the same color should be considered as in the same group, and hence encoded by the same color. Hence the color mapping strategy in this situation should also be carefully considered because each region on the Munsell card has its color: we should not violate it too much. Thus in this design, the fundamental principle of color encoding is to ensure that the colored region shares a similar color to that in the original Munsell card.

Before generating the figure, we should first count the term(s) used for each chip and fill Table 6. In the results collected from one language, it is possible for one chip to have multiple terms. Like for the chip B01,

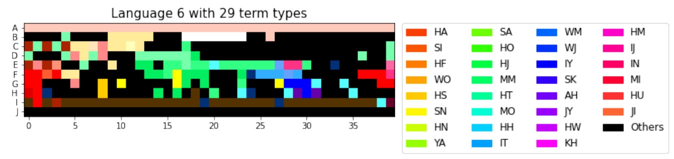


Fig. 5. Bad example of allocating unused term to the unrecorded chips when all the terms are highlighted.

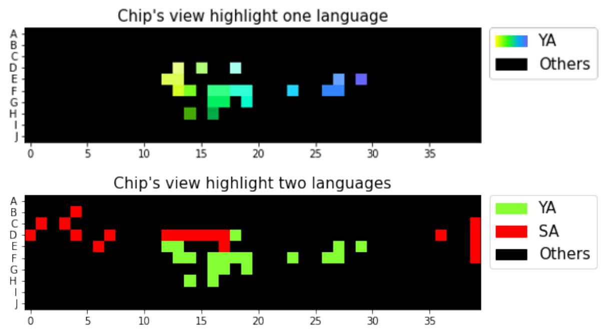


Fig. 6. Good examples of allocating unused term to the unrecorded chips when only several terms are highlighted.

there are 25 participants who use GG and 3 who use GA. Under this condition, we choose the term used by the majority (i.e., GG for B01) as the selected term, similar to what the authors of [7, 9, 11, 14] did in their work.

Table 6. Counts of term(s) for different chips, the number in the bracket behind term is the number of participant.

Chip ID	...	B01	B02	B03	...
Selected Term	...	GG	?	AG	...
Term(s)	...	GG(25)	-	AG(28)	...
	...	GA(3)	-	-	...

Following the design described above, we might come up with a figure like Fig 5. In this figure, we allocate different colors to the chips on the Munsell based on the group they are in (i.e., which term represents them). The chips in the same group are allocated the same color. For the chips with no records in the dataset, we put them in the 'others' group and color them black. This visualization is designed to assist the researchers to find patterns of the language. However, honestly, this figure is a bit messy and it is hard for us to conclude any pattern from them. There are two main reasons for this:

- **The number of groups (i.e., different term types) is too large, which makes the figure too crowded and some distinct groups have too similar colors;**
- **There are so many chips with no records (in 'other' class). Some region(s) of the same color are split apart by those black chips, which make these regions not continuous any more. The user might consider these split regions as different clusters, even though in fact they are in the same group.**

To counter these problems, we improve our design in the following two directions.

Reducing the groups:

As mentioned in Chapter 13 of [10], a reduction is necessary when the visualization is too complex: a concise figure would highlight the most important information for the users. Plus, the researchers also admit that sometimes they prefer to focus on one or a few terms each time to obtain a clearer view. Hence we propose two good examples in Figure 6, which only highlight one or two groups on one figure. In the upper panel of Figure 6, we only highlight one term using a Munsell color encoding method, i.e., all chips represented by 'YA' is colored by their Munsell color while all other chips (including the chips with no records) are colored black. From this figure, it is easy to see that 'YA' might represent yellow, green, and blue with normal brightness. We also provide an example of highlighting two groups in the bottom panel of Figure 6. Under this condition, we can no longer use the Munsell color encoding method, because that would make the user confuse about the border between two groups. Hence we select the major color of each term, i.e., the color of the chip with the most recordings in this language. In this example, we allocate red to 'SA' and green to 'YA'. From this figure, it is clear that term 'YA' and 'SA' is tangled for some regions, e.g., the region from $D15$ to $H20$ (the square region with $D15$ to $H20$ as the diagonal). Plus, in this language, it is very likely that the second term (i.e., 'A') denotes the brightness while the first term (i.e., 'Y' and 'S') denote the hue. The researchers can select other terms, e.g., 'XA' (if exist in this language) to verify their hypothesis.

Merging the groups and interpolating chips:

Another important requirement mentioned in Task 2.1 is the visualization of compositionality of the whole language. From the description in Task 2.1, we know that a high compositional language should satisfy the following two requirements:

- One attribute always represents the hue and another attribute always represents the brightness;
- The same letter should represent the same attribute in most of the terms, e.g., 'X' always represents red in 'XA', 'XB' and 'XM'.

We can definitely use the vis idiom like Figure 6 to show the compositionality of **some** terms in one language, but sometimes the researchers care more about the compositionality of the **whole** language, which means we need other methods to solve the two conundrums mentioned before (i.e., too many groups and the chips with no records). Inspired by the discussion in Chapter 13.4 in [10], combining the definition of compositionality and the adjacency principle, we propose to use group merging and chips interpolating methods together.

Note that the group merging and chips interpolation both have two directions. If we want to observe the compositionality from the hue's perspective, we would merge all the terms starting with the same letter to the same group, e.g., 'GA', 'GG', 'GL' and 'GK' are merged to 'G*'. At the same time, we would put the unrecorded term to specific groups using the adjacency principle. For example, in Figure 7, the chip D2 is in group 'G*' and D9 is in group 'M*', the chip D4 and D8 are unrecorded. If we interpolate D4 and D8 using hue's perspective, we would put D4 in 'G*' because D4 is closer to D2 than D9. Similarly, we put D8 in 'M*'. If we want to observe the structure from the brightness's perspective, we merge the terms ending with the same letter to the same group, e.g., 'GA', 'SA', 'MA', and 'QA' to '*A'. For example, in Figure 7, the chip D2 is in group '*A' and H2 is in group '*K'. Then we put E2 and G2 into '*A' and '*K' respectively.

With two methods mentioned above, we demonstrate a high and a low compositional language in Figure 8, from hue's perspective (the upper panels) and brightness's perspective (the bottom panels). To give a better intuition on what would a purely compositional language be like in our vis idiom, we also propose examples of perfect language (i.e., the Munsell hue-brightness coordinate expression) on the rightmost column in Figure 8. Another design for our vis idioms is color-coding. Different from the strategy used in Figure 6, we do not simply use the color selected from the Munsell card. Rather, for the examples from hue's perspective, we use colors with different hues (but the same brightness). For the examples from brightness's perspective, we use colors with different brightness (but the same hue). Such a color encoding can make the researchers focus more on the structure in one direction in one figure: it is clear that most of the borders between different groups in the high compositional language are parallel with the axis, while the borders of low compositional language are more convoluted.

4.2 Design from Term's View

From the principles mentioned in Chapter 11 in [10], manipulating the view of the data in the figure can provide additional information.

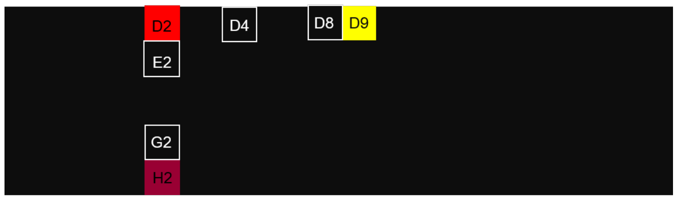


Fig. 7. Explain what is interpolation.

Besides, in Task 2.2, the researchers are curious about the expressivity (i.e., how many concepts one word represents) of each term. In the WCS dataset, the expressivity on one term can be represented by the number of chips that term is mapped to. At the same time, they might be anxious that the chip interpolation using the adjacent principle mentioned before can introduce bias. Under such a condition, a vis idiom from the term's view is helpful. Compared with the chip's view we mentioned above, the biggest difference in term's view is that we no longer need to 'guess' the term for those unrecorded chips, which makes our figure unbiased to the real data.

Before drawing the figure, we also need to count the chip(s) represented by each term in the data. Similarly, we should fill Table 7. Note that in this table, each used term represents at least one chip. Besides, it is also possible that one chip is represented by multiple terms.

Table 7. Counts of chip(s) for different terms, the number in the bracket behind term is the number of participant.

Term	...	GG	GA	AG	...
Chip(s)	...	B01(25)	B04(28)	B03(22)	...
	...	B02(3)	-	-	...

Based on the principles mentioned in Chapter 7.3 in [10], we list all the possible terms used in this language, followed by all the chips they represent, as illustrated by the example in Figure 9. To provide a better intuition of what color each chip represents, we use small rectangles with the color copied from the Munsell card and align these rectangles to make the figure easier to read. The width of these rectangles are the same, so the entire length of all the rectangles followed can represents the number of chips that one term represents, and hence represents the expressivity of that term.

4.3 Discussions of Visualizing a Single Language

The tasks discussed in this section focus more on the result of a single language because the researchers expect to get an overview of the characteristics (e.g., compositionality, expressivity, etc) of each language. From the definitions of these characteristics, we believe the best solution in this part is to illustrate the results from different perspectives, i.e., chip's view and term's view. That is because the geometric information of the chips plays a crucial role in demonstrating compositionality while the total number of chips is the most important feature for expressivity. It is hard to put both of them in the same figure: they might distract the user's attention and introduce illusion.

Another interesting design for this part is the color-coding method for different chips. Most of the time, it would be nice to use a color map that represents the true color of specific chips on the Munsell card. However, the color map fixing the hue (or the brightness) can also achieve a reasonable result in specific situations, like the examples in Figure 8.

5 Solutions for Visualizing Quantitative Measurements

A quantitative metric is necessary for most of the research fields because the researchers can then have standard evaluation criteria to compare different models or observe the correlation among multiple attributes. Such a requirement also exists in the fields studying WCS dataset. Based on the discussions with the fieldworkers, we know that the main requirement for Task 3.1 is the correlation among multiple quantitative metrics. Plus, they also want to see the distribution of these metrics, because the distribution can implicitly tell them more about the hidden generating factor behind the observed data.

Table 8. What-why-how analysis for Task 3.1.

What	Derived quantitative metrics: topological similarity, inconsistency and term types
Why	Find correlation among multiple metrics. Observe the distribution of the metrics.
How	Use scatter plot to demonstrate two metrics together. Put linear regression line on top of the scatter plot. Use histogram and envelope curve for distributions. Let histogram share axis of the scatter chart. Use mark size and color encode another attribute.

To demonstrate the correlation between two metrics, we chose the scatter plot as a starting point. As illustrated in the examples in Figure 10, we chose two attributes then use x-value and y-value to encode them. To better demonstrate the linear correlation among the two attributes, we draw a line chart for linear regression and use the shadow region to represent the 90% confidence interval. Following the principles provided in Chapter 12 in [10], we superimpose the line chart over the scatter plot because they share the same coordinate system, and drawing them in the same region can provide a good intuition of how the scatter points disperse along the regression line. Furthermore, we also juxtapose the two marginal distributions on top of (or to the right of) the main plot, sharing the corresponding axis. Such a design can achieve a higher information density, and can also help the user to see whether the points are densely or sparsely distributed in a specific region. From these examples, we can conclude that the topological similarity and term types are positively linear correlated. The topological similarity follows a Gaussian-like distribution and the inconsistency might follow an exponential-like distribution.

If the researchers want to observe the correlations among three attributes in one figure, we might use the color or size channel of spots to encode this attribute. For example, in Figure 11, the x-axis and y-axis represent two attributes. The color channel (together with the size channel) encodes another attribute using different levels of hue. In this design, it is only possible to draw a regression line between the value encoded by two axes. Hence it is better to allocate the color channel to the unimportant attributes.

In summary, the combination of scatter a plot and linear a regression line is suitable for demonstrating the linear correlation between two attributes. The color or size channel can also be used to encode more attributes to incorporate more information into the same figure. Plus, the axis sharing method (e.g., sharing between the scatter plot and the line chart, sharing between scatter and histogram) is also a good way to enhance information density in a vis idiom. However, we should also keep in mind that the users are more sensitive to some encoding methods than others. So we should use the most significant channel to highlight the most important feature.

6 Discussion and Future Work

In this section, we will first introduce how we implement our visualization designs in this paper. Then a brief overview of the related work is provided. The limitations and potential future work is also discussed.

6.1 Implementation

As the author is not familiar with JavaScript (and D3), we select two common packages in Python to implement all the visualization designs mentioned above, i.e., matplotlib[2] and seaborn[3]. The project is released on Github (`https://github.com/Joshua-Ren/CPSC547_YIREN`). The files and the descriptions of them are provided in Table 9. We chose to use Jupyter Notebook[4], which is an open-source web application that allows us to create and share documents that contain live code,

[2]The website of this package is: `https://matplotlib.org`
[3]The website of this package is `https://seaborn.pydata.org`
[4]The website of this is `https://jupyter.org/`

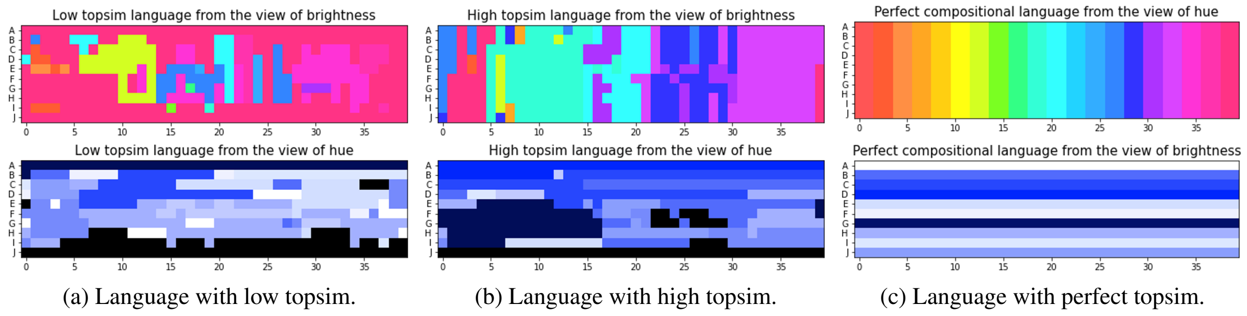


Fig. 8. Examples of the pattern of languages with different topological similarity in two views. The upper panels are generated from the view of hue while the bottom panels are from the view of brightness.

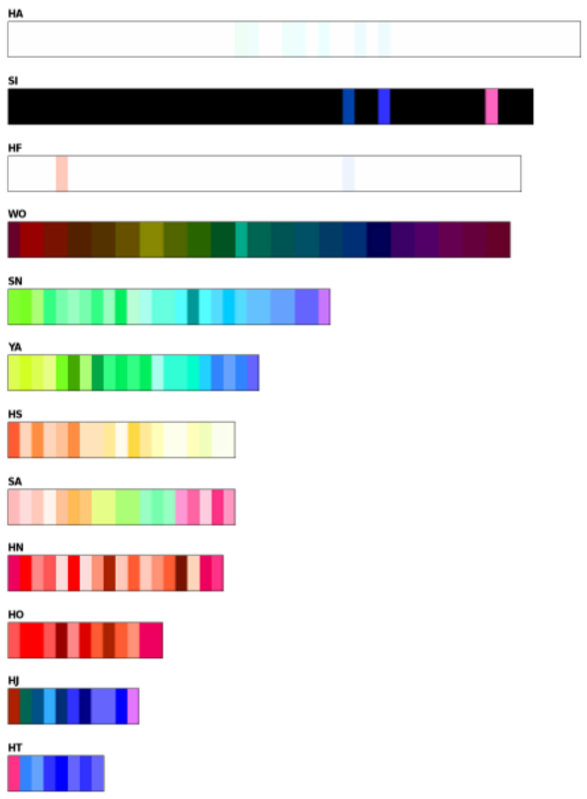


Fig. 9. Examples of term's view, ordered by the expressivity.

equations, visualizations, and narrative text. I put part of my software code in the appendix to demonstrate how could the user modify the code to generate new figures. As the readers can download the source code and modify any settings as they wish: they can use it as a starting point of an interactive design, or directly draw their own figures and put them on their report.

6.2 Related Work

As the focus of this paper is visualizing the raw WCS dataset, we mainly compare the corresponding part in the related works. That is because, in most of these works, the authors would not just visualize the raw WCS dataset: they might calculate many statistics to support their hypotheses. Among the related works, most of the visualization is similar to the examples in Figure 8, as in Figure 1 and 2B in [9], Figure 4 in [14] and Figure 1 in [7]. The difference between our design and theirs is that we merge the terms and hence are capable of showing all terms in one figure. Plus, the color encoding we applied here is also different from theirs. Furthermore, in Figure 2A in [9], the authors also

Table 9. Files in the released project, ** is used to avoid the table being too long.

spkr.txt	Raw data from WCS dataset
foci-exp.txt	Raw data from WCS dataset
Munsell.txt	Mapping from chip to RGB color
demograph**.ipynb	Codes for distribution of participants
single**.ipynb	Codes for visualizing single language
quantitative**.ipynb	Codes for visualizing quantitative metrics

provide a series of pie charts to show the diversity of all 110 languages, which is quite similar to our Figure 2. To the best of our knowledge, there are not so many figures demonstrating demographic information in these works. That is probably because the authors have already selected their interesting languages before. The design of the term's view is also quite novel among all the related works.

6.3 Limitations and Future Work

Due to the time constraints, we only complete the code for the most fundamental function of our visualization design, i.e., our code can only generate static images. Obviously, it would be nice to derive an interactive visualization for the users to get intuitions easier. For example, we can let the users arbitrarily select the language, age range, or gender of the participants they wish to observe, in any visualization designs we mentioned in the paper. However, such a design needs more powerful software tools, e.g., D3, which is left to our future work.

7 Conclusions

The WCS dataset is widely applied in the fields of evolutionary linguistics, cognitive science, and other related directions. To assist the fieldworkers in getting intuition from the raw dataset, we first interview with some experts and then conclude five typical tasks. After that, based on the principles provided in VAD, we provide several visualization designs for each task. The designs cover three aspects of the dataset, i.e., the demographic information of participants, the pattern of a single language, and the correlation among multiple quantitative metrics. We also implement all the designs using common packages in python. With the help of our work, the researchers can flexibly change parameters and generate figures as they need.

References

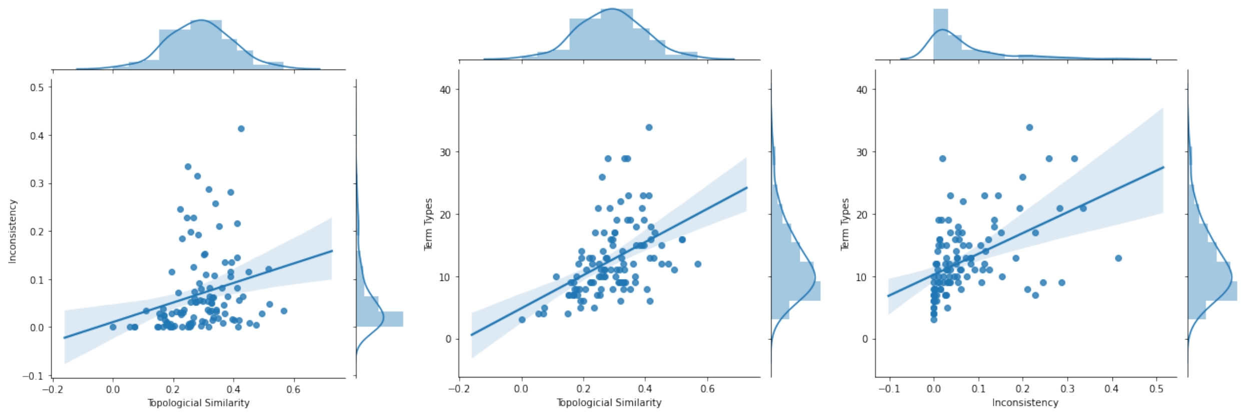
[1] J. Andreas. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019.

[2] H. Brighton and S. Kirby. Understanding linguistic evolution by visualizing the emergence of topographic mappings. *Artificial life*, 12(2):229–242, 2006.

[3] T. Briscoe. *Linguistic evolution through language acquisition*. Cambridge University Press, 2002.

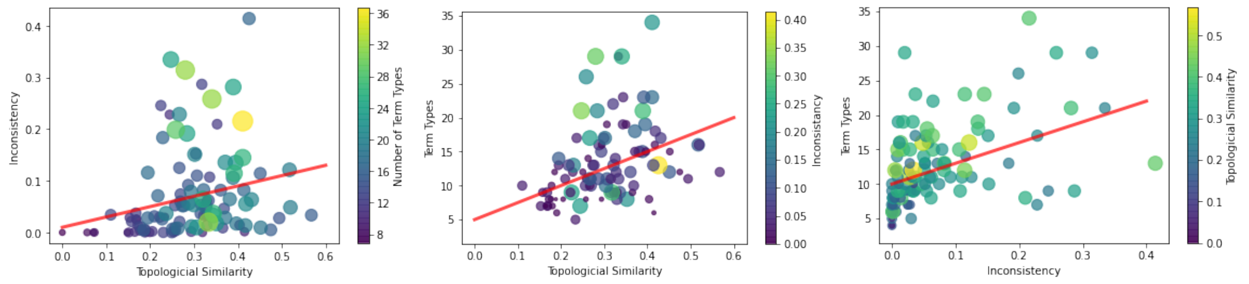
[4] H. H. Clark. *Using language*. Cambridge University Press, 1996.

[5] M. Cogswell, J. Lu, S. Lee, D. Parikh, and D. Batra. Emergence of compositional language with deep generational transmission. *arXiv preprint arXiv:1904.09067*, 2019.



(a) Topsim v.s. incons. (b) Topsim v.s. term types. (c) Incons v.s. term types.

Fig. 10. Joint plot of scatter plot, linear regression and distribution.



(a) Topsim v.s. incons. (b) Topsim v.s. term types. (c) Incons v.s. term types.

Fig. 11. Using size and color channel to encode another attribute together.

[6] M. R. Costa-jussà. An analysis of gender bias studies in natural language processing. *Nature Machine Intelligence*, pp. 1–2, 2019.

[7] E. Gibson, R. Futrell, J. Jara-Ettinger, K. Mahowald, L. Bergen, S. Ratnasingam, M. Gibson, S. T. Piantadosi, and B. R. Conway. Color naming across languages reflects color use. *Proceedings of the National Academy of Sciences*, 114(40):10785–10790, 2017.

[8] S. Kirby, M. Tamariz, H. Cornish, and K. Smith. Compression and communication in the cultural evolution of linguistic structure. *Cognition*, 141:87–102, 2015.

[9] D. T. Lindsey and A. M. Brown. World color survey color naming reveals universal motifs and their within-language diversity. *Proceedings of the National Academy of Sciences*, 106(47):19785–19790, 2009.

[10] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[11] T. Regier, P. Kay, and N. Khetarpal. Color naming reflects optimal partitions of color space. *Proceedings of the National Academy of Sciences*, 104(4):1436–1441, 2007.

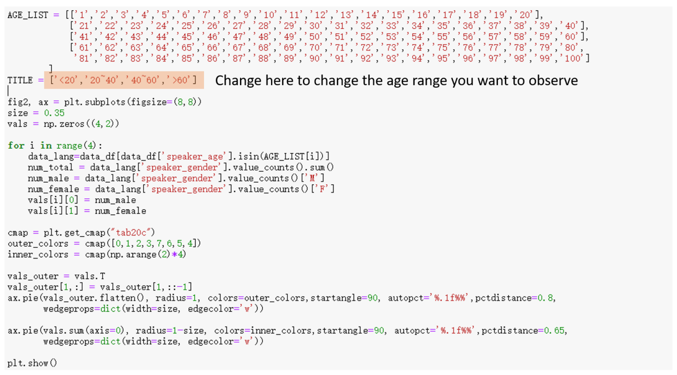
[12] T. Regier, C. Kemp, and P. Kay. Word meanings across languages support efficient communication. *The handbook of language emergence*, 87:237, 2015.

[13] P. K. Richard Cook and T. Regier. WCS data archives. *http://www1.icsi.berkeley.edu/wcs/data.html*.

[14] N. Zaslavsky, C. Kemp, T. Regier, and N. Tishby. Efficient compression in color naming and its evolution. *Proceedings of the National Academy of Sciences*, 115(31):7937–7942, 2018.

8 Appendix: Example Code for This Project

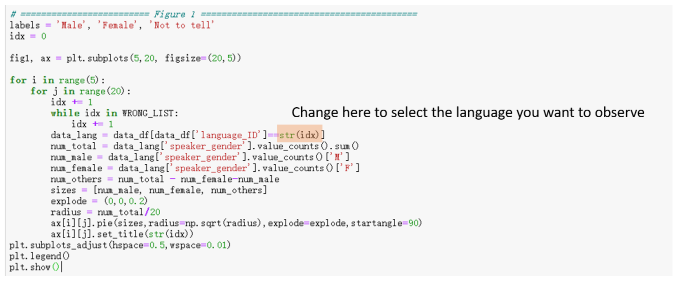
To help the users better apply our visualization design, we propose some examples in this appendix. Actually, there are more functions which are not shown here in our project, the user can modify any part of our code to generate figures of their own style.



```
AGE_LIST = [['1','2','3','4','5','6','7','8','9','10','11','12','13','14','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','43','44','45','46','47','48','49','50','51','52','53','54','55','56','57','58','59','60'],
            ['61','62','63','64','65','66','67','68','69','70','71','72','73','74','75','76','77','78','79','80',
             '81','82','83','84','85','86','87','88','89','90','91','92','93','94','95','96','97','98','99','100']
        ]
TITLE = ['<20','20~40','40~60','>60']
|
fig2, ax = plt.subplots(figsize=(8,8))
size = 0.35
vals = np.zeros((4,2))

for i in range(4):
    data_lang=data_df[data_df['speaker_age'].isin(AGE_LIST[i])]
    num_total = data_lang['speaker_gender'].value_counts().sum()
    num_male = data_lang['speaker_gender'].value_counts()['M']
    num_female = data_lang['speaker_gender'].value_counts()['F']
    vals[i][0] = num_male
    vals[i][1] = num_female

cmap = plt.get_cmap("tab20c")
outer_colors = cmap([0,1,2,3,7,6,5,4])
inner_colors = cmap(np.arange(2)*4)

vals_outer = vals.T
vals_outer[1,:] = vals_outer[1,::-1]
ax.pie(vals_outer.flatten(), radius=1, colors=outer_colors,startangle=90, autopct='%.1f%%',pctdistance=0.8,
       wedgeprops=dict(width=size, edgecolor='w'))

ax.pie(vals.sum(axis=0), radius=1-size, colors=inner_colors,startangle=90, autopct='%.1f%%',pctdistance=0.65,
       wedgeprops=dict(width=size, edgecolor='w'))

plt.show()
```

Change here to change the age range you want to observe

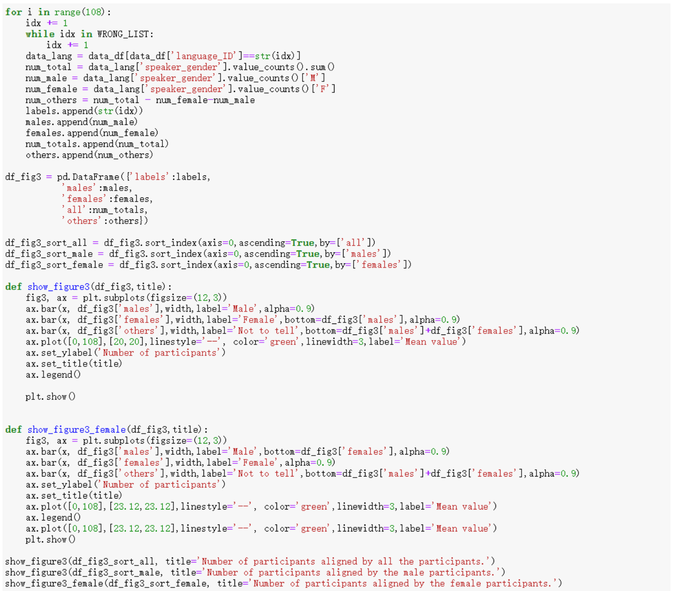
Fig. 12. Code for generating pie-ring charts. The user can modify the age range by changing the highlight part.



```
# ======================== Figure 1 ========================================
labels = 'Male', 'Female', 'Not to tell'
idx = 0

fig1, ax = plt.subplots(5,20, figsize=(20,5))

for i in range(5):
    for j in range(20):
        idx += 1
        while idx in WRONG_LIST:
            idx += 1
        data_lang = data_df[data_df['language_ID']==str(idx)]
        num_total = data_lang['speaker_gender'].value_counts().sum()
        num_male = data_lang['speaker_gender'].value_counts()['M']
        num_female = data_lang['speaker_gender'].value_counts()['F']
        num_others = num_total - num_female-num_male
        sizes = [num_male, num_female, num_others]
        explode = (0,0,0.2)
        radius = num_total/20
        ax[i][j].pie(sizes,radius=np.sqrt(radius),explode=explode,startangle=90)
        ax[i][j].set_title(str(idx))
plt.subplots_adjust(hspace=0.5,wspace=0.01)
plt.legend()
plt.show()|
```

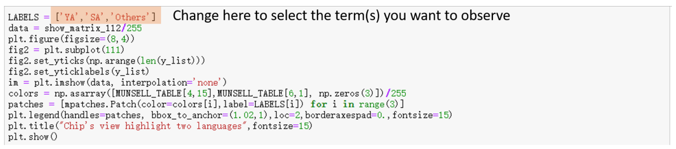
Change here to select the language you want to observe

Fig. 13. Code for generating aligned pie charts. The user can modify what languages they want to observe by changing the highlight part.



```python
for i in range(108):
    idx += 1
    while idx in WRONG_LIST:
        idx += 1
    data_lang = data_df[data_df['language_ID']==str(idx)]
    num_total = data_lang['speaker_gender'].value_counts().sum()
    num_male = data_lang['speaker_gender'].value_counts()['M']
    num_female = data_lang['speaker_gender'].value_counts()['F']
    num_others = num_total - num_female-num_male
    labels.append(str(idx))
    males.append(num_male)
    females.append(num_female)
    num_totals.append(num_total)
    others.append(num_others)

df_fig3 = pd.DataFrame({'labels':labels,
           'males':males,
           'females':females,
           'all':num_totals,
           'others':others})

df_fig3_sort_all = df_fig3.sort_index(axis=0,ascending=True,by=['all'])
df_fig3_sort_male = df_fig3.sort_index(axis=0,ascending=True,by=['males'])
df_fig3_sort_female = df_fig3.sort_index(axis=0,ascending=True,by=['females'])

def show_figure3(df_fig3,title):
    fig3, ax = plt.subplots(figsize=(12,3))
    ax.bar(x, df_fig3['males'],width,label='Male',alpha=0.9)
    ax.bar(x, df_fig3['females'],width,label='Female',bottom=df_fig3['males'],alpha=0.9)
    ax.bar(x, df_fig3['others'],width,label='Not to tell',bottom=df_fig3['males']+df_fig3['females'],alpha=0.9)
    ax.plot([0,108],[20,20],linestyle='--', color='green',linewidth=3,label='Mean value')
    ax.set_ylabel('Number of participants')
    ax.set_title(title)
    ax.legend()

    plt.show()


def show_figure3_female(df_fig3,title):
    fig3, ax = plt.subplots(figsize=(12,3))
    ax.bar(x, df_fig3['males'],width,label='Male',bottom=df_fig3['females'],alpha=0.9)
    ax.bar(x, df_fig3['females'],width,label='Female',alpha=0.9)
    ax.bar(x, df_fig3['others'],width,label='Not to tell',bottom=df_fig3['males']+df_fig3['females'],alpha=0.9)
    ax.set_ylabel('Number of participants')
    ax.set_title(title)
    ax.plot([0,108],[23.12,23.12],linestyle='--', color='green',linewidth=3,label='Mean value')
    ax.legend()
    ax.plot([0,108],[23.12,23.12],linestyle='--', color='green',linewidth=3,label='Mean value')
    plt.show()

show_figure3(df_fig3_sort_all, title='Number of participants aligned by all the participants.')
show_figure3(df_fig3_sort_male, title='Number of participants aligned by the male participants.')
show_figure3_female(df_fig3_sort_female, title='Number of participants aligned by the female participants.')
```

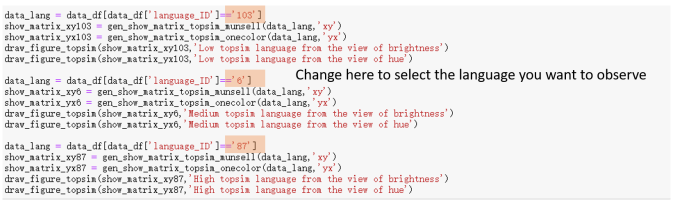
Fig. 14. Code for generating stacked bar chart.



```python
LABELS = ['YA','SA','Others']
data = show_matrix_112/255
plt.figure(figsize=(8,4))
fig2 = plt.subplot(111)
fig2.set_yticks(np.arange(len(y_list)))
fig2.set_yticklabels(y_list)
im = plt.imshow(data, interpolation='none')
colors = np.asarray([MUNSELL_TABLE[4,15],MUNSELL_TABLE[6,1], np.zeros(3)])/255
patches = [mpatches.Patch(color=colors[i],label=LABELS[i]) for i in range(3)]
plt.legend(handles=patches, bbox_to_anchor=(1.02,1),loc=2,borderaxespad=0.,fontsize=15)
plt.title("Chip's view highlight two languages",fontsize=15)
plt.show()
```

Change here to select the term(s) you want to observe

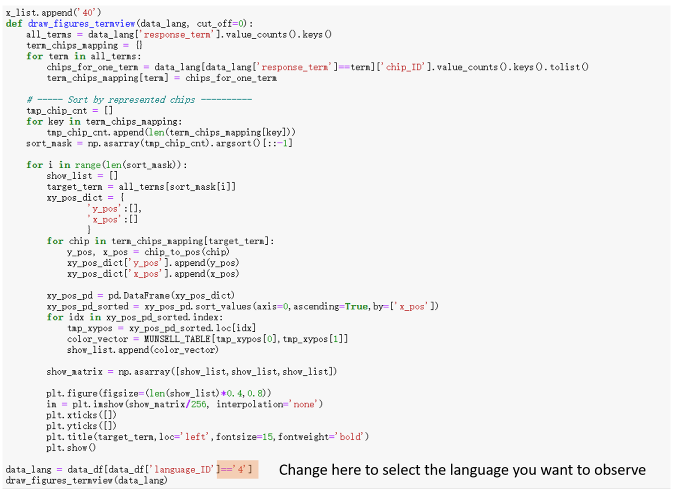
Fig. 15. Code for showing one or two terms from chip's view. The user can modify the lighted part to select which term they want to observe.



```python
data_lang = data_df[data_df['language_ID']=='103']
show_matrix_xy103 = gen_show_matrix_topsim_munsell(data_lang,'xy')
show_matrix_yx103 = gen_show_matrix_topsim_onecolor(data_lang,'yx')
draw_figure_topsim(show_matrix_xy103,'Low topsim language from the view of brightness')
draw_figure_topsim(show_matrix_yx103,'Low topsim language from the view of hue')

data_lang = data_df[data_df['language_ID']=='6']
show_matrix_xy6 = gen_show_matrix_topsim_munsell(data_lang,'xy')
show_matrix_yx6 = gen_show_matrix_topsim_onecolor(data_lang,'yx')
draw_figure_topsim(show_matrix_xy6,'Medium topsim language from the view of brightness')
draw_figure_topsim(show_matrix_yx6,'Medium topsim language from the view of hue')

data_lang = data_df[data_df['language_ID']=='87']
show_matrix_xy87 = gen_show_matrix_topsim_munsell(data_lang,'xy')
show_matrix_yx87 = gen_show_matrix_topsim_onecolor(data_lang,'yx')
draw_figure_topsim(show_matrix_xy87,'High topsim language from the view of brightness')
draw_figure_topsim(show_matrix_yx87,'High topsim language from the view of hue')
```

Change here to select the language you want to observe

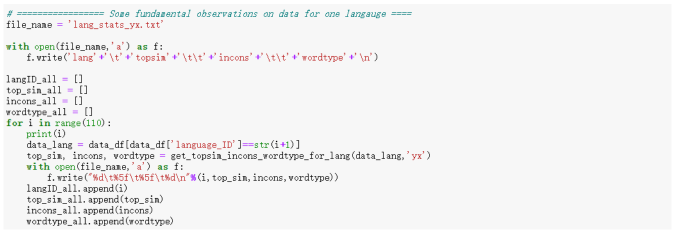
Fig. 16. Code for showing all terms with the help of group merging and chip interpolation. The code for implementing these two mechanisms is not shown here. The user can modify the lighted part to select which languages they want to observe.



```python
x_list.append('40')
def draw_figures_termview(data_lang, cut_off=0):
    all_terms = data_lang['response_term'].value_counts().keys()
    term_chips_mapping = {}
    for term in all_terms:
        chips_for_one_term = data_lang[data_lang['response_term']==term]['chip_ID'].value_counts().keys().tolist()
        term_chips_mapping[term] = chips_for_one_term

    # ----- Sort by represented chips ----------
    tmp_chip_cnt = []
    for key in term_chips_mapping:
        tmp_chip_cnt.append(len(term_chips_mapping[key]))
    sort_mask = np.asarray(tmp_chip_cnt).argsort()[::-1]

    for i in range(len(sort_mask)):
        show_list = []
        target_term = all_terms[sort_mask[i]]
        xy_pos_dict = {
                'y_pos':[],
                'x_pos':[]
                }
        for chip in term_chips_mapping[target_term]:
            y_pos, x_pos = chip_to_pos(chip)
            xy_pos_dict['y_pos'].append(y_pos)
            xy_pos_dict['x_pos'].append(x_pos)

        xy_pos_pd = pd.DataFrame(xy_pos_dict)
        xy_pos_pd_sorted = xy_pos_pd.sort_values(axis=0,ascending=True,by=['x_pos'])
        for idx in xy_pos_pd_sorted.index:
            tmp_xypos = xy_pos_pd_sorted.loc[idx]
            color_vector = MUNSELL_TABLE[tmp_xypos[0],tmp_xypos[1]]
            show_list.append(color_vector)

        show_matrix = np.asarray([show_list,show_list,show_list])

        plt.figure(figsize=(len(show_list)*0.4,0.8))
        im = plt.imshow(show_matrix/256, interpolation='none')
        plt.xticks([])
        plt.yticks([])
        plt.title(target_term,loc='left',fontsize=15,fontweight='bold')
        plt.show()

data_lang = data_df[data_df['language_ID']=='4']
draw_figures_termview(data_lang)
```

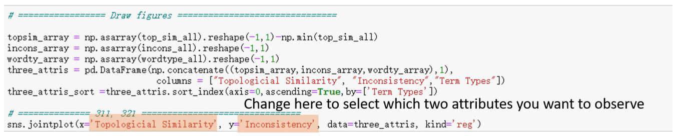
Change here to select the language you want to observe

Fig. 17. Code for showing one single language from term's view. The user can modify the lighted part to select which language they want to observe.



```python
# ================= Some fundamental observations on data for one langauge ====
file_name = 'lang_stats_yx.txt'

with open(file_name,'a') as f:
    f.write('lang'+'\t'+'topsim'+'\t\t'+'incons'+'\t\t'+'wordtype'+'\n')

langID_all = []
top_sim_all = []
incons_all = []
wordtype_all = []
for i in range(110):
    print(i)
    data_lang = data_df[data_df['language_ID']==str(i+1)]
    top_sim, incons, wordtype = get_topsim_incons_wordtype_for_lang(data_lang,'yx')
    with open(file_name,'a') as f:
        f.write("%d\t%5f\t%5f\t%d\n"%(i,top_sim,incons,wordtype))
    langID_all.append(i)
    top_sim_all.append(top_sim)
    incons_all.append(incons)
    wordtype_all.append(wordtype)
```

Fig. 18. Code for reading the statistics of the whole dataset. How to do such calculation can be found in the .ipynb file. The user can chose which languages they want to consider for the next step.



```python
# ================= Draw figures ===============================

topsim_array = np.asarray(top_sim_all).reshape(-1,1)-np.min(top_sim_all)
incons_array = np.asarray(incons_all).reshape(-1,1)
wordty_array = np.asarray(wordtype_all).reshape(-1,1)
three_attris = pd.DataFrame(np.concatenate((topsim_array,incons_array,wordty_array),1),
                            columns = ["Topologicial Similarity", "Inconsistency","Term Types"])
three_attris_sort =three_attris.sort_index(axis=0,ascending=True,by=['Term Types'])

# ============== 311, 321 ================================
sns.jointplot(x='Topologicial Similarity', y='Inconsistency', data=three_attris, kind='reg')
```

Change here to select which two attributes you want to observe

Fig. 19. Code for drawing scatter plot, linear regression line and histogram. The user can modify the lighted part to select which attributes they want to compare.