

Programming Data Visualizations that Show Data Composition

DVP

CS5 (*continued*)

PPTX Companion to CS5 video
called:

*Activity - Experiment
with R and Python
Code for DVP to Show
Data Composition*

Data Visualizations – Data Composition

- How to program data visualization that show data composition in R.
- How to program data visualizations that show data composition in Python.



Part-to-a-whole



Visualisation methods that show part (or parts) of a variable to it's total. Often used to show how something is divided up.



Donut Chart



Marimekko Chart



Pie Chart



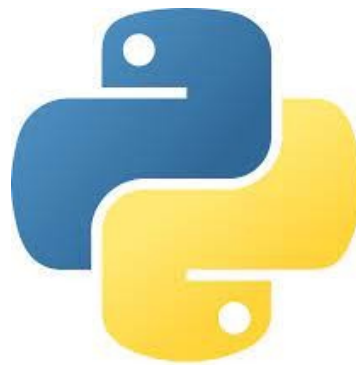
Stacked Bar Graph



Sunburst Diagram



Treemap



Part-to-a-whole

Visualisation methods that show part (or parts) of a variable to it's total. Often used to show how something is divided up.



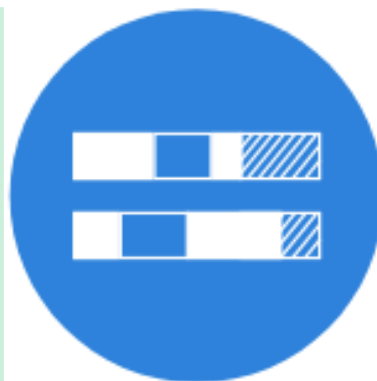
Donut Chart



Marimekko Chart



Pie Chart



Stacked Bar Graph



Sunburst Diagram



Treemap

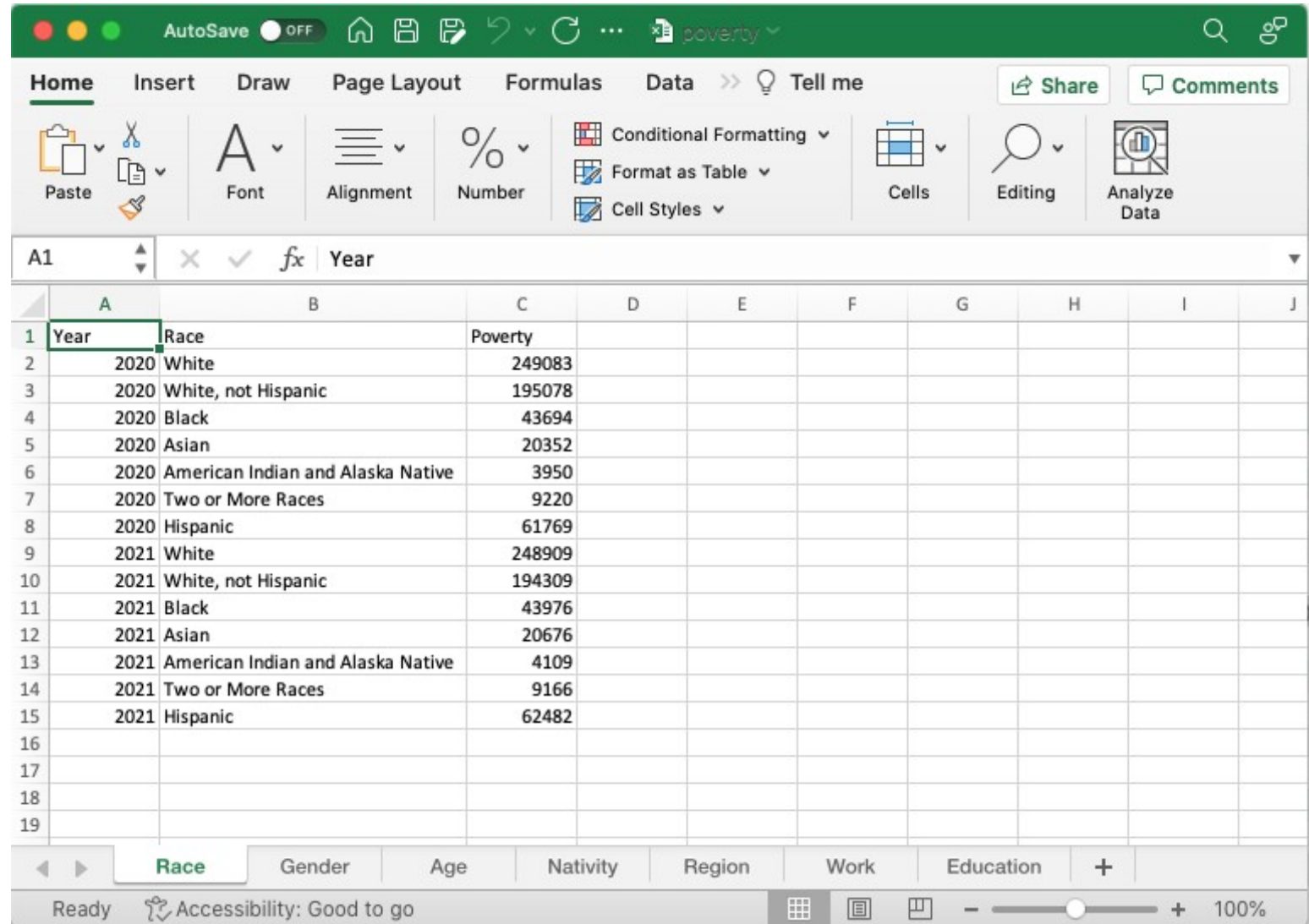
Class Session 5 Activity Data

Get the file (xlsx) from the CS5 Bb folder.

Unit of measurement:

- count in thousands (people living in poverty)

Objective: Assess the data composition based on some population characteristics called *demographics*



The screenshot shows a Microsoft Excel spreadsheet with the following data:

Year	Race	Poverty
2020	White	249083
2020	White, not Hispanic	195078
2020	Black	43694
2020	Asian	20352
2020	American Indian and Alaska Native	3950
2020	Two or More Races	9220
2020	Hispanic	61769
2021	White	248909
2021	White, not Hispanic	194309
2021	Black	43976
2021	Asian	20676
2021	American Indian and Alaska Native	4109
2021	Two or More Races	9166
2021	Hispanic	62482

The spreadsheet interface includes the following elements:

- Top Bar:** AutoSave OFF, Home, Insert, Draw, Page Layout, Formulas, Data, Tell me, Share, Comments.
- Formulas Bar:** A1, Year.
- Columns:** A (Year), B (Race), C (Poverty), D, E, F, G, H, I, J.
- Rows:** 1 to 19.
- Bottom Bar:** Ready, Accessibility: Good to go, 100% zoom.

Our process for programming data visualizations in **R**.

- Create an R *Markdown* document (.Rmd) "R codebook"
- Insert code and documentation lines and optimize them as we go along
- Install packages if required
- Load libraries
- Import data
- Explore the data
- Do some data processing
- Define variables
- Create additional variables if needed
- Create data visualizations (DVs)
- Save DVs as stand-alone image files
- Insert DVs into a report document and write data insights
- Download completed R codebook



Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:
<https://login.rstudio.cloud/login?redirect=%2F>

At this point,

- locate the **.Rmd codebook** that you created as a result of engaging in the previous Class Session 4 activity that Dr. Mead called, **"Class_Session_4_Activity-DVP-LS24.Rmd"** in her demonstration contained in the CS4 video called, *"Activity - Experiment with R and Python Code for DVP to Show How Data Change Over Time"*, then
- follow what Dr. Mead demonstrates to do next after this slide point in the CS5 video, *"Activity - Experiment with R and Python Code for DVP to Show Data Composition"*.

Poverty XLSX - data composition in R



```
# Create Data Visualizations (DVs) that show Data Composition in R using an XLSX file.  
# Dataset source: U.S. Census Bureau, Current Population Survey, 2021 and 2022 Annual Social and Economic Supplements (CPS ASEC).  
https://www.census.gov/topics/income-poverty/poverty/data/datasets.html  
# The data used in this Activity are people living in poverty (count in thousands).
```

```
if(!require("readxl")) install.packages("readxl")  
if(!require("tidyverse")) install.packages("tidyverse")  
if(!require("fs")) install.packages("fs")  
if(!require("scales")) install.packages("scales")  
if(!require("ggplot2")) install.packages("ggplot2")
```

```
library(readxl)  
library(tidyverse)  
library(fs)  
library(scales)  
library(ggplot2)
```

```
read_excel("Class_Session_5_Activity.xlsx")
```

```
# The excel_sheets() function can extract the names of the sheets after you assign the dataset to a variable.  
path <- "Class_Session_5_Activity.xlsx"  
path %>%  
  excel_sheets()
```

```
# To read just one sheet to a dataframe  
data <- read_excel("Class_Session_5_Activity.xlsx", 1) # "1" will load the first sheet in the xlsx file. Experiment with changing "1" to another  
available sheet number that you want to look at.
```

```
head(data)
```

```
view(data)
```

(R code continued on next slide with "R2")

Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:

<https://login.rstudio.cloud/login?redirect=%2F>

R1

Poverty XLSX – data composition in R



Read more about all above from the sources: <https://dominicroye.github.io/en/2019/import-excel-sheets-with-r/>
and <https://rveryday.wordpress.com/2016/11/29/examine-a-data-frame-in-r-with-7-basic-functions/>

Do some data engineering to make separate dataframes for years 2020 and 2021

```
data_2020 <- subset(data, data$Year==2020)
```

```
data_2021 <- subset(data, data$Year==2021)
```

```
view(data_2020)
```

```
view(data_2021)
```

Do some data engineering to define some variables and to create some additional new variables for potential use later. (We might not actually use them in this Activity, but It's good practice.)

```
ethnicity20 <- data_2020$Ethnicity
```

```
poverty20 <- data_2020$Poverty
```

```
sum_p_2020 <- sum(data_2020$Poverty)
```

```
ethnicity21 <- data_2020$Ethnicity
```

```
poverty21 <- data_2021$Poverty
```

```
sum_p_2021 <- sum(data_2021$Poverty)
```

```
percent_p_2020 <- percent(poverty20/sum_p_2020)
```

```
percent_p_2021 <- percent(poverty21/sum_p_2021)
```

(R code continued on next slide with "R3")

R2

Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:

<https://login.rstudio.cloud/login?redirect=%2F>

Poverty XLSX – data composition in R



Create a customized pie DV to show the composition of the poverty data for the year 2020.

`library(ggplot2)` # Use the "library()" function to load a library called "ggplot2" for creating data visualizations.

```
g <- ggplot(data_2020, aes(x = "", y = poverty20, fill = ethnicity20)) + geom_col(color = "white") + ggtitle("2020 U.S. Poverty by Ethnicity") + geom_text(aes(x = 1.5, label = percent_p_2020), position = position_stack(vjust = 0.5)) + coord_polar("y") + scale_fill_brewer(palette = 'Set1') + theme_void() + theme(plot.title = element_text(hjust = 0.5)) + theme(legend.title=element_blank())
```

`g` # Call the "g" variable created in a previous step to show the data visualization (DV) on the screen.

If your computer screen is small, you will have an issue with the the percentage data labels in the pie DV overlapping each other.

Try recreating the pie DV with the addition of a package called "ggrepel" to manipulate the labels.

```
if(!require("ggrepel")) install.packages("ggrepel")
library(ggrepel)
```

```
g <- ggplot(data_2020, aes(x = "", y = poverty20, fill = ethnicity20)) +
  geom_col(color = "white") +
  ggtitle("2020 U.S. Poverty by Ethnicity") +
  geom_label_repel(aes(label = percent_p_2020), position = position_stack(vjust = 0.5), show.legend = FALSE) +
  coord_polar("y") +
  scale_fill_brewer(palette = 'Set1') +
  theme_void() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.title = element_blank())
```

`g` # Call the revised "g" variable created in a previous step to show the revised data visualization (DV) on the screen.

(R code continued on next slide with "R4")

Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:

<https://login.rstudio.cloud/login?redirect=%2F>

R3

Poverty XLSX – data composition in R



If optimal, use the "ggsave()" function to save your DV as a stand-alone image asset (file).

`ggsave("Poverty_by_Ethnicity_2020-pie-R.jpeg")` # Experiment with creating different stand-alone image asset file formats, e.g., .png or .jpg or .pdf

Alternative for creating stand-alone image asset file in the RStudio Cloud web-based IDE: Expand the boundaries of the "Plots" pane in the RStudio Cloud web-based IDE until the labels of your DV are clearly visible and not overlapping one another. Then click "export" and save your chart as an image asset (.png file).

Experiment with setting different colors on your R DV: <https://ggplot2-book.org/scale-colour.html>

Learn how to customize your R DV here: <https://r-graph-gallery.com/index.html>

Create a copy the code for the 2020 year pie chart on this slide **and edit** it in all relevant parts to create a pie DV for poverty by the same demographic that you used for **2021**. This will allow for visual comparisons between the two data compositions.

Don't forget to download your completed .Rmd codebook for later reuse.

Experiment with repeating the above code sections (R1, R2, R3) to visualize a different sheet (variable) in the dataset, such as Gender, Age, Nativity, Region, Work, Education. Note: this will require you to make several edits in the code to get it to work for that specific variable.

(R code continued on next slide with "R5")

R4

Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:

<https://login.rstudio.cloud/login?redirect=%2F>

Poverty XLSX – data composition i



Experiment with making a donut DV for 2020 by repeating the code for a pie chart but "cutting a hole" in the center of it.

Create test data by hand for this session, and we will experiment with optimization in a future session.

```
data <- data.frame(  
  category = c("White", "White, not Hispanic", "Black", "Asian", "American Indian and Alaska Native", "Two or More Races", "Hispanic"),  
  count = c(42.71, 33.45, 7.49, 3.49, 0.68, 1.58, 10.59)  
)
```

```
data$fraction <- data$count / sum(data$count) # To compute percentages  
data$ymax <- cumsum(data$fraction) # To compute cumulative percentages to set a top boundary  
data$ymin <- c(0, head(data$ymax, n = -1)) # To compute the bottom boundary  
data$labelPosition <- (data$ymax + data$ymin) / 2 # To compute data label position  
data$label <- paste0(data$category, "\nValue: ", round(data$count, 1), "%") # To display the label with percentage sign  
  
explode <- c(0, 0, 0, -0.6, -0.05, -0.52, 0) # Define an explode effect
```

```
ggplot(data, aes(ymax = ymax, ymin = ymin, xmax = 4 + explode, xmin = 3 + explode, fill = category)) +  
  geom_rect() +  
  geom_label(aes(x = 4 + explode, y = labelPosition, label = label), size = 4) +  
  scale_fill_brewer(palette = "Set3") +  
  coord_polar(theta = "y") +  
  xlim(c(2, 4)) +  
  theme_void() +  
  ggtitle("2020 U.S. Poverty by Ethnicity") +  
  theme(legend.position = "none",  
        plot.title = element_text(hjust = 0.5, size = 18)) # Center the plot title
```

```
ggsave("Poverty_by_Ethnicity_2020-donut-R.png", width = 7, height = 7, dpi = 300) # Experiment with image asset format, size and quality.
```

(R code continued on next slide with "R6")

R5

Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:

<https://login.rstudio.cloud/login?redirect=%2F>

Poverty XLSX – data composition in R



Experiment with making a basic horizontal bar DV with the data.

```
New_Colors <- c('green', 'blue', 'purple', 'brown', 'black', 'orange', 'gray') # Data engineer a new variable to define some colors that to use in the DV.
```

```
# Experiment with the different colors to customize your DV.
```

```
g <- ggplot(data_2020, aes(x = poverty20, y = reorder(ethnicity20, poverty20))) + # To order the ethnicity categories by highest (on top) to lowest (on bottom) on the DV
```

```
  geom_bar(stat = "identity", fill = New_Colors) + # Use your "New_Colors" variable here
```

```
  labs(title = "2020 U.S. Poverty by Ethnicity", x = "People Living in Poverty", y = "") + # To remove the unnecessary y-axis label
```

```
  theme_minimal() +
```

```
  theme(axis.text.y = element_text(size = 10)) + # To adjust y-axis text size
```

```
  scale_x_continuous(labels = comma) + # To add commas to the numbers on the x-axis
```

```
  guides(fill = FALSE) # To remove legend for fill colors
```

```
g + coord_flip() # To flip coordinates to create horizontal bars for optimal viewing
```

```
g # Call the variable to show the DV
```

Don't forget to download your completed .Rmd codebook for later reuse.

(end of R code set)

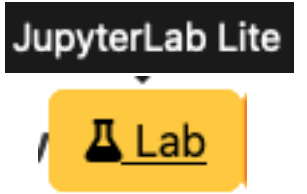
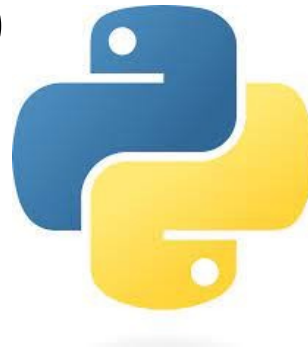
R6

Students in this course are **required** to do R exercises in the RStudio Cloud web-based environment:

<https://login.rstudio.cloud/login?redirect=%2F>

Our process for programming data visualizations in **Python**.

- Create Python notebook (.ipynb)
- Insert code and documentation lines and optimize them as we go along
- Import libraries (install items as needed)
- Import data
- Explore the data
- Do some data processing
- Define variables
- Create additional variables if needed
- Create data visualizations (DVs)
- Save DVs as stand-alone image files
- Insert DVs into a report document and write data insights
- Download completed Python notebook

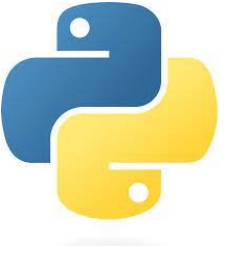


Students in this course are **required** to do Python exercises in the JupyterLite web-based environment:
<https://jupyter.org/try-jupyter/lab/>

At this point,

- locate the **.ipynb codebook** that you created as a result of engaging in the previous Class Session 4 activity that Dr. Mead called, "**Class_Session_4_Activity-DVP-LS24.ipynb**" in her demonstration contained in the CS4 video called, "*Activity - Experiment with R and Python Code for DVP to Show How Data Change Over Time*", then
- follow what Dr. Mead demonstrates to do next after this slide point in the CS5 video, "*Activity - Experiment with R and Python Code for DVP to Show Data Composition*".

Poverty XLSX – data composition in Python



Create Data Visualizations (DVs) that show Data Composition in Python using an XLSX file.

Dataset source: U.S. Census Bureau, Current Population Survey, 2021 and 2022 Annual Social and Economic Supplements (CPS ASEC).

<https://www.census.gov/topics/income-poverty/poverty/data/datasets.html>

The data used in this Activity are people living in poverty (count in thousands).

```
import piplite
await piplite.install("openpyxl")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

P1

Use the "ExcelFile()" function from a library called "panda" that was imported as a variable called "pd" to read an excel file called "Class_Session_5.xlsx" and each of its sheets and save it to a variable called "xlsx" for later use.

```
xlsx = pd.ExcelFile('Class_Session_5.xlsx')
```

Use the "sheet_names" attribute to output a list all sheets in the an xlsx file that was assigned to a variable called "xlsx".

```
xlsx.sheet_names
```

If want to, read more about the above xlsx multiple-sheet data inspection steps from the source: <https://stackoverflow.com/questions/26521266/using-pandas-to-pd-read-excel-for-multiple-worksheets-of-the-same-workbook>

Use the "read_excel()" function from a library called "panda" that was imported as a variable called "pd" to read just one sheet to a dataframe assigned to a variable called "df".

```
df = pd.read_excel(xlsx, sheet_name="Ethnicity")
```

 # Experiment with calling in different sheet names from the list of sheets output from the previous step.

Call the "df" variable to see what the data for that sheet looks like

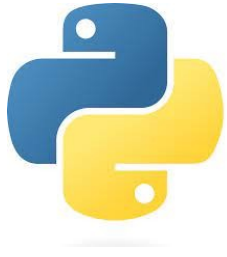
```
df
```

(Python code continued on next slide with "P2")

Students in this course are **required** to do Python exercises in the JupyterLite web-based environment:

<https://jupyter.org/try-jupyter/lab/>

Poverty XLSX – data composition in Python



```
# Do some data engineering to make separate dataframes for years 2020 and 2021
```

```
df_2020 = df[df.Year == 2020]
```

```
df_2021 = df[df.Year == 2021]
```

```
df_2020
```

```
df_2021
```

```
# Do some data engineering to define some variables and to create some additional new variables for potential use later. (We might not actually use them in this Activity, but It's good practice.)
```

```
ethnicity20 = df_2020['Ethnicity']
```

```
poverty20 = df_2020['Poverty']
```

```
ethnicity21 = df_2021['Ethnicity']
```

```
poverty21 = df_2021['Poverty']
```

```
explode = [0, 0, 0.2, 0.2, 0.2, 0.2, 0.2] # An "explode" list variable is useful in some DVs for when you want to make certain data of the composition to stand out to the viewer. The number of components in your explode list variable must match the number of categories of the variable that you are visualizing. Experiment with the values later to see how it changes the look of your DV.
```

P2

Students in this course are **required** to do Python exercises in the JupyterLite web-based environment:

<https://jupyter.org/try-jupyter/lab/>

(Python code continued on next slide with "P3")

Poverty XLSX – data composition in Python



```
# Create a customized pie DV to show the composition of the poverty data for the year 2020.
fig, ax = plt.subplots(figsize=(12, 7), subplot_kw=dict(aspect="equal"), dpi= 80)
plt.pie(poverty20, autopct='%0f%%', explode=explode) # To make sure it shows percentages and to use your "explode" variable.
plt.title("2020 U.S. Poverty by Ethnicity")
plt.ylabel("") # To make sure that ylabel is blank because we don't need one this time.
plt.xlabel("") # To make sure that xlabel is blank because we don't need one this time.
plt.legend(ethnicity20, loc='upper left', framealpha=0.25) # Variations on legend placement and properties to enhance viewability when
needed
plt.savefig("Poverty_by_Ethnicity_2020-pie-Python.png") # Use the "savefig()" function from the module called "pyplot" from the library
called "matplotlib" that was imported as a variable called "plt" to to save your DV as a stand-alone image asset (file) (.png or .jpg, etc.).
plt.show()
```

P3

Create a copy the code for the 2020 year pie chart on this slide **and edit** it in all relevant parts to create a pie DV for poverty by the same demographic that you used for **2021**. This will allow for visual comparisons between the two data compositions.
Don't forget to download your completed .ipynb codebook for later reuse.

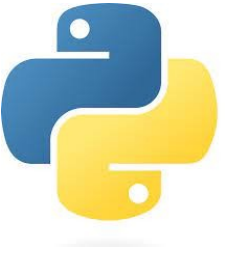
Experiment with customizing your Python DV: <https://indianaiproduction.com/matplotlib-pie-chart/>
Experiment with repeating the above code sections (P1, P2, P3) to visualize a different sheet (variable) in the dataset, such as Gender, Age, Nativity, Region, Work, Education.
Note: this will require you to make several edits in the code to get it to work for that specific variable.
For example, one edit is that you will have to edit the number of items in the explode set to match the number of variable types.

(Python code continued on next slide with "P4")

Students in this course are **required** to do Python exercises in the JupyterLite web-based environment:

<https://jupyter.org/try-jupyter/lab/>

Poverty XLSX – data composition in Python



Experiment with making a donut DV for 2020 by repeating the code for a pie chart but "cutting a hole" in the center of it.

```
fig, ax = plt.subplots(figsize=(12, 7), subplot_kw=dict(aspect="equal"), dpi= 80)
```

```
plt.pie(poverty20, autopct='%0f%%', explode=explode)
```

```
centre_circle = plt.Circle((0, 0), 0.70, fc='white') # draw the circle
```

```
fig.gca().add_artist(centre_circle) # Add the circle into the pie
```

```
fig = plt.gcf()
```

```
plt.title("2020 U.S. Poverty by Ethnicity")
```

```
plt.ylabel("")
```

```
plt.xlabel("")
```

```
plt.legend(ethnicity20, loc='center left')
```

#Hint: Insert code line here to save your DV as a stand-alone image asset file with an optimized filename.

```
plt.show()
```

Don't forget to download your completed .ipynb codebook for later reuse.

(Python code continued on next slide with "P5")

P4

Students in this course are **required** to do Python exercises in the JupyterLite web-based environment:

<https://jupyter.org/try-jupyter/lab/>

Poverty XLSX – data composition in Python



Experiment with making a basic horizontal bar DV with the data.

`New_Colors = ['green', 'blue', 'purple', 'brown', 'teal', 'orange', 'gray']` # Data engineer a new variable to define some colors that to use in the DV.
Experiment with the different colors to customize your DV.

```
plt.barh(ethnicity20, poverty20, color=New_Colors) # User your "New_Colors" variable here.  
plt.title('2020 U.S. Poverty by Ethnicity')  
plt.ylabel('')  
plt.xlabel('People Living in Poverty')  
#Hint: Insert code line here to save your DV as a stand-alone image asset file with an optimized filename.  
plt.show()
```

Don't forget to download your completed .ipynb codebook for later reuse.

(end of Python code set)

P5

Students in this course are **required** to do Python exercises in the JupyterLite web-based environment:

<https://jupyter.org/try-jupyter/ab/>

End of file.