SOC-GA 2332 Intro to Stats Lab 4

Risa Gelles-Watnick

9/26/2025

```
knitr::opts_chunk$set(echo = TRUE, fig.pos = "H", out.extra = "")

pacman::p_load(
  tidyverse,
  kableExtra,
  ipumsr, # for reading IPUMS data into R
  ggplot2
)
```

Part 1: Reviewing P-values

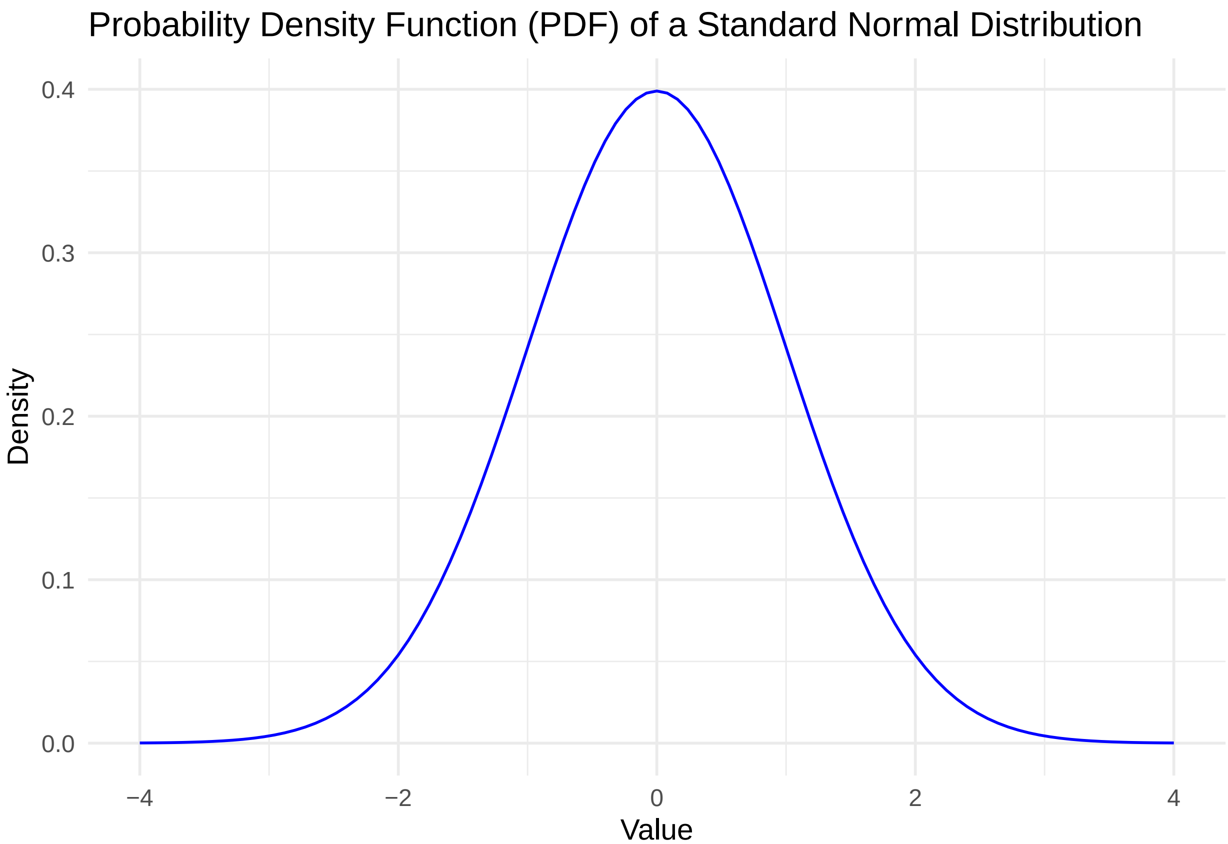
- p-value is the probability that the test statistic equals to (or is more extreme than) what we observed
- **Probability Density Function (PDF):** a function that gives the probability of getting each value of a continuous variable on a random draw.

$$\text{PDF}(x) = P(X = x)$$

```
# plotting a PDF of a standard normal distribution

# creating a data frame with an x-axis range for the plot
plot_data <- data.frame(x = c(-4, 4))

# Plot the normal distribution PDF
ggplot(data = plot_data, aes(x = x)) +
  stat_function(fun = dnorm,  # telling it to plot every value of the density function for the normal d
                args = list(mean = 0, sd = 1), # providing the parameters for a standard normal distrib
                color = "blue", # setting the color of the line
                ) +
  labs(title = "Probability Density Function (PDF) of a Standard Normal Distribution",
       x = "Value",
       y = "Density") +
  theme_minimal() # applying a clean theme for the plot
```



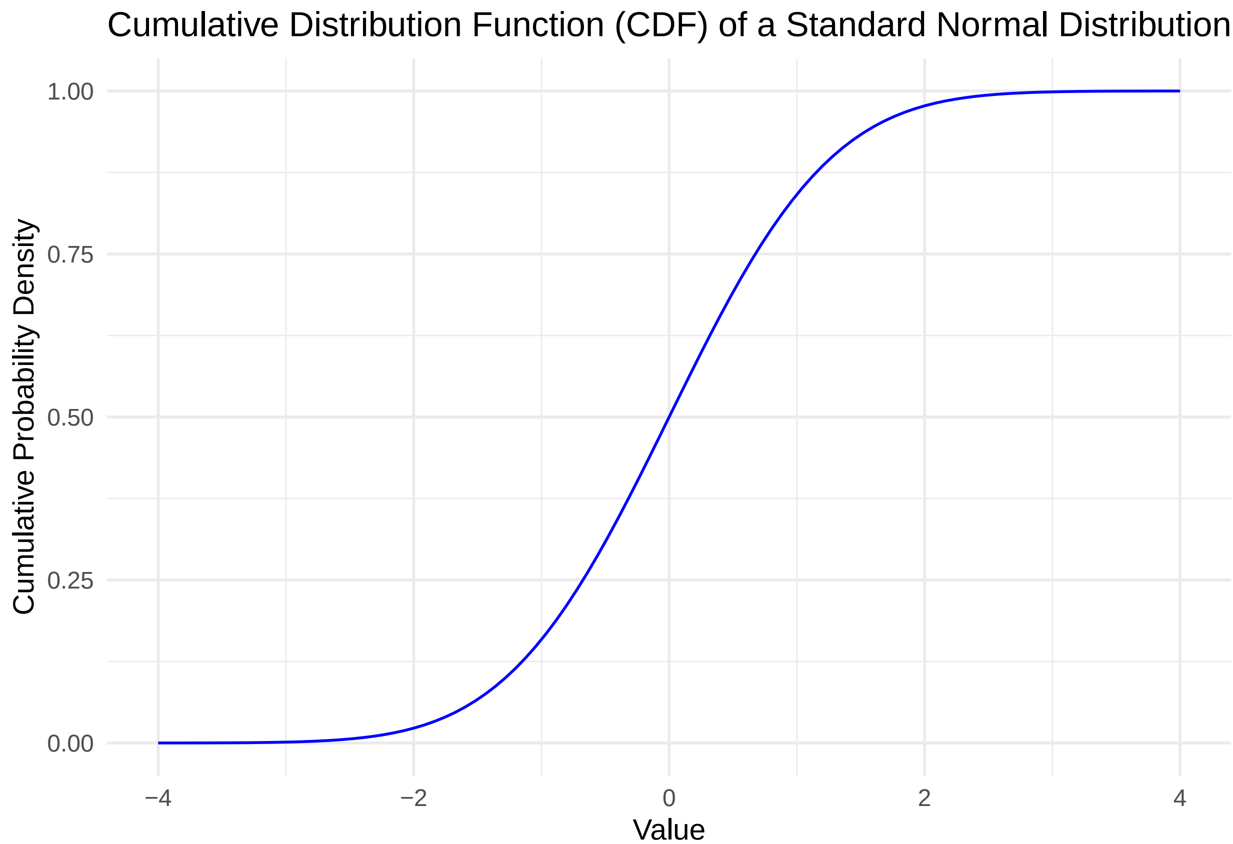
-
  - What does the area under any PDF sum to?
  - Why can't we just use the value of the PDF as our p-value?
- The integral of this function (i.e. all the area under the curve from point A to B) gives us the **Cumulative Distribution Function (CDF)**, which is the probability at each value of a continuous variable that if you take a random draw from the variable, you will get a value less than or equal to that value.

$$\mathrm{CDF}(x) = P(X \leq x) = \int_{-\infty}^{x} \mathrm{PDF}(t)dt$$

```
# plotting a CDF of a standard normal distribution

# creating a data frame with an x-axis range for the plot
plot_data <- data.frame(x = c(-4, 4))

# Plot the normal distribution PDF
ggplot(data = plot_data, aes(x = x)) +
  stat_function(fun = pnorm,  # telling it to plot every value of the density function for the normal d
                args = list(mean = 0, sd = 1), # providing the parameters for a standard normal distrib
                color = "blue", # setting the color of the line
                ) +
  labs(title = "Cumulative Distribution Function (CDF) of a Standard Normal Distribution",
       x = "Value",
       y = "Cumulative Probability Density") +
  theme_minimal() # applying a clean theme for the plot
```



- Last week we introduced the `pt()` function for calculating p-values in R. Let's break down what the `pt()` does in more detail.
  - `p` is for probability and `t` is for t-test.
  - This function gives the value of the cumulative distribution function (or the integral of the probability density function) at the point you specify.
  - In the `pt()` function, you enter your degrees of freedom (the only parameter needed for a t-test) and a test parameter value, then the function generates a t-distribution (which assumes null hypothesis and is therefore centered around 0) and calculates the value of the CDF for that distribution at your test statistic value.
  - `pt(q = observed_t, df = your_degree_of_freedom, lower.tail = FALSE)` calculates the probability of randomly drawing a value on the null t distribution with your degrees of freedom that is between your observed test statistic and positive infinity. The `lower.tail = FALSE` argument tells it to calculate the upper tail rather than the lower tail, i.e. your test statistic or higher rather than your negative test statistic and lower
  - Why do we multiply this function by two when we're doing a two-tailed test? `2*pt(q = observed_t, df = your_degree_of_freedom, lower.tail = FALSE)`.
  - When we get to regressions next week, the `lm()` command in R will do a lot of this work automatically for you, but the same calculations of the PDF and CDF still apply.

Part 2: IPUMS!

Exercise 1: Download Data from IPUMS

1. Register an account for using IPUMS-USA here

2. Once your account is created, log in to IPUMS-USA and create a data extract request by selecting the **2010 ACS** sample and the **SEX**, **EDUC**, and **INCWAGE** variables. You can keep the preselected variables. Your "Data Cart" should be similar to this:

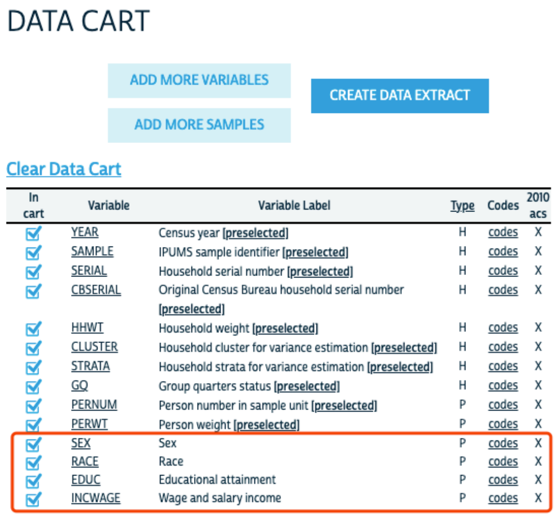


Figure 1: Data Cart

3. To save time, create a small data extract with a customized sample size of **0.01%**. As shown in the following screenshot. Submit your data extract request.

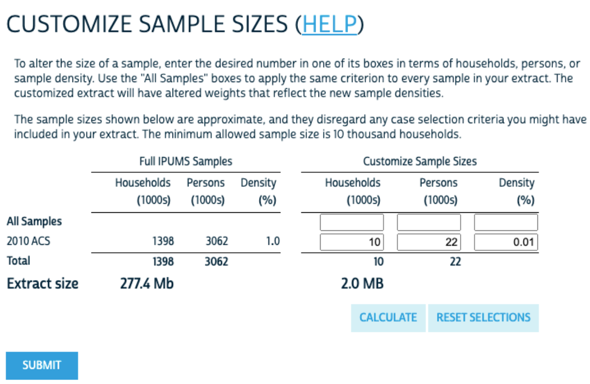


Figure 2: Customize

4. While IPUMS is processing our data extract request, you can prepare the file needed for loading the IPUMS data to R. Open the "DDI" page in your browser, it should be a `.xml` page. Save this `.xml` file to your current lab folder, or you can save it in a sub-folder devoted to data files, such as a sub-folder called "data" within your current lab folder.

5. Once your data is ready on IPUMS, you will receive an email with a download link. (The wait time varies from a couple minutes to a couple days, depending on how large your files are and how busy the server is.) Follow the link in your email and download your data (the default format is `.dat.gz`). **Make sure to save it at the same directory with your `.xml` file**

6. Unzip the `.gz` data file using your computer's unzip applications (for Mac users, if the default extractor doesn't work, try the Unarchiver application). Now you should have the `.dat` file and the `.xml` file in the same directory.

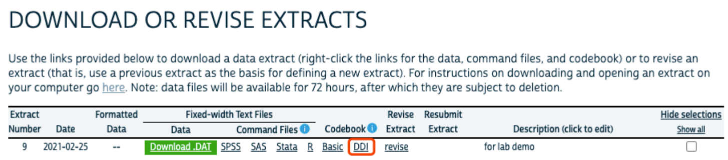


Figure 3: Download

7. Load the data to your R environment, by using the code shown in the webpage when you click the `R` Command File link in your IPUMS data downloading page.

Figure 4: Download 2

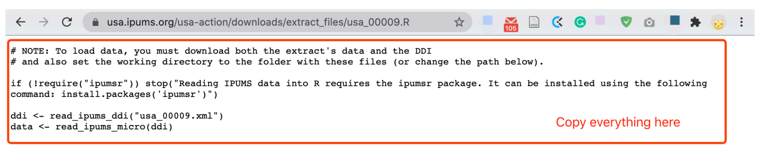


Figure 5: Command

```r
# paste code here
```

Reading IPUMS Documentations

Before working with the data, we should be clear about how the variables are coded. You can access the documentation of each variables through the `.xml` webpage you opened earlier.

When you have that page open in your browser, click "Variable Description" to access the codebook for each variables we just downloaded.

The other way is to read the documentation of each variable directly in the online system. This is my preferred approach.

Exercise 2: Reading the Data Codebook (15 minutes)

Go through the variable description, and answer the following question:

1. What does the variable "PERNUM" represents? How can you uniquely identify each person within the IPUMS with this variable?
2. What does the variable "PERWT" represents? When should you consider using this variable?
3. For the variable "SEX", what are the possible values it can take? What does each value represent? Try run `str(data$SEX)`, what do you see?
4. What does the variable "EDUC" represents? How many values this variable can take? What is the value that represents N/A?
5. For the variable "INCWAGE", what are the codes for N/A and missing data?

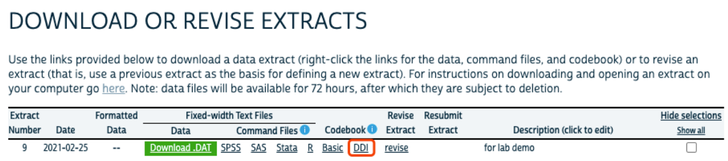


Figure 6: Documentation 1

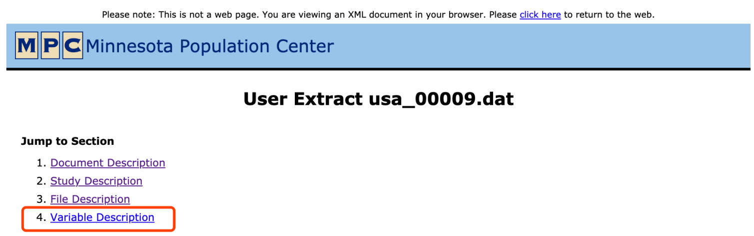


Figure 7: Documentation 2

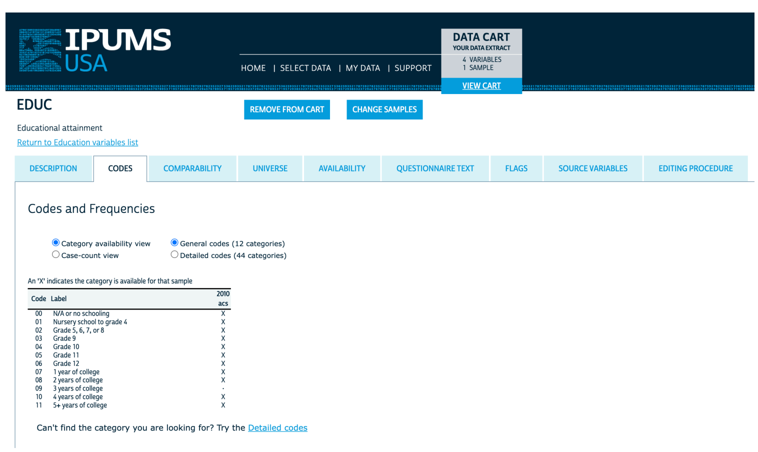


Figure 8: Codebook

Exercise 3: Running a t-test with IPUMS data (20 minutes)

We will try to run a t-test on the average income in highly educated people versus people with lower levels of formal education. So we will be using the **education and wage income** variables. (Normally we would do this exercise with regressions, but since we haven't done that lecture yet we'll do the same data cleaning steps but then run a t-test instead). First, we need to make sure we know how the variables are distributed in the sample and remove N/A and missing values (we do not consider imputing missing values at this point).

Note: Since we each extracted a 0.01% random sample, it is normal if your results are slightly different from this demo.

1. First, make sure your data is loaded in your environment.

```
# your code here
```

2. Then, create a new dataset that keeps only the following variables for subsequent data analysis: SAMPLE, EDUC, INCWAGE

```
# your code here
```

3. Removing missing values

Since we are running a t-test with **education and wage income**, we need to remove observations that do not have any of the two values. You can either remove then by filtering out the values that represent N/A or missing based on the variable codebook, or you can recode these values to `NA` so that you keep all the observations. For this example we will do the former, but we will cover the latter method in the future.

First, check the coding scheme for each variable of interest in the codebook. Then, filter out the missing values. Last, check the new distributions for each variable with a table or plot.

```
# your code here
```

4. Now, let's try to estimate whether the difference in income between highly educated people and people with less education is statistically significant. First, create two education groups: one of people who completed high school or less (Grade 12 or lower) and one of people who completed 4 years of college or more. You can do this by creating two new datasets or by creating an indicator variable in the existing dataset.

```
# your code here
```

5. Make a plot showing the difference in means between these groups. Feel free to use any plot format you think best shows this difference.

```
# your code here
```

6. Now run a t-test to see if the difference in income between the high and low educated groups is statistically significant. Explain what the meaning of the p-value is (using the ideas we covered at the beginning of lab). Interpret both the statistical and substantive significance of the results.

```
# your code here
```