

## 1.3: Creating a New Variable and Producing Summary Statistics

### Uniform Crime Report

This chapter will teach us how to create a new variable and produce summary statistics. We will use the crime data from the Uniform Crime Report (UCR)—specifically, the 2018 Part 1 crime data from Pennsylvania. More than 18,000 police departments in the US report crime data to the FBI, and the FBI compiles the nationwide data and publishes the UCR.

Crime analysts need to know the differences between Part 1 crime and Part 2 crime. There are two categories of criminal offenses. Part 1 offenses, known as index crimes or serious crimes, are generally felonies that can result in more than a year of incarceration in prisons. Some violent and property crimes that fall under Part 1 offenses include homicide, rape, robbery, aggravated assault, burglary, larceny-theft, motor vehicle theft, and arson. On the other hand, Part 2 offenses are non-index crimes and are considered less serious. These crimes are generally misdemeanors that warrant less than a year of incarceration. Part 2 offenses may include simple assault, vandalism, fraud, drug offenses, disorderly conduct, and other misdemeanors. The biggest limitation of the UCR is that the data only counts crimes reported to the police, so you cannot know how many crimes are underreported. This unreported crime is known as dark or hidden figures of crime, and several data-collecting strategies have been developed to estimate this underreporting. I will introduce these datasets in different chapters. In this chapter, we will learn how to compute descriptive statistics using Part 1 crime data.

### Readxl Package

First, you will need to download the file labeled “2018.UCR.PA.xlsx” from [the shared Google Drive folder containing the 2018.UCR.PA.xlsx data](#). The data we will use is from an Excel file. The haven package may be less suitable for importing an Excel file (typically with an xlsx extension) than it was for opening the SPSS file we used previously. So, we should first install and load the readxl package. Then, we will read the file into R.

```
install.packages("readxl")
library(readxl)
X2018.UCR.PA <- read_excel("C:/Users/75JCH0I/OneDrive - West Chester
University of PA/WCU Research/R/data/2018.UCR.PA.xlsx")
View(X2018.UCR.PA)
```

This code loads the readxl package and then reads an Excel file at the specified path. The contents of this file are then assigned to an object named X2018.UCR.PA. It's worth noting that I added an “X” before the object's name “2018.UCR.PA” because R doesn't allow a number to be the first character in an object name.

When you use the view function to inspect the data, you'll notice that the dataset's structure differs from the 2012 General Social Survey (GSS) data we worked with in the previous chapter. In the GSS dataset, each row represented a respondent, and each column represented a variable. However, in the UCR dataset, each row represents a city. There are 12 variables in total, including city and population.

The variable “violent crime” in the dataset is computed by aggregating all four different types of Part 1 violent offenses: murder/involuntary manslaughter, rape, robbery, and aggravated assault. Similarly, the variable “property crime” is computed by summing up all four different types of Part 1 property offenses: burglary, larceny-theft, motor vehicle theft, and arson.

### Rename Function

Please use the summary function to see if the variables are coded properly.

```
summary(X2018.UCR.PA)
```

The results show that there are some issues with the way variables were labeled. For example, you may notice that the variable name is “Violent\r\ncrime”. This occurs because of how the data is formatted in the Excel file. There was a space between “Violent” and “crime” in the Excel file. We need to rename these variables so that variable names are simplified and easy to type. Load the tidyverse package and use the rename function from the dplyr package.

```
library(tidyverse)
```

```
# Assuming 'my_data' is your data frame or tibble and you want to change the variable
name from 'old_name' to 'new_name'
my_data <- my_data %>%
  rename(new_name = old_name)
#Renaming Violent\r\ncrime to violent.crime
X2018.UCR.PA.cleaned <- X2018.UCR.PA %>%
  rename(violent.crime = Violent\r\ncrime)
```

Unfortunately, the code above did not work because R was confused about managing backslash characters in the variable name. When a variable name contains special characters such as backslashes, it is important to escape them in R properly.

```
X2018.UCR.PA.cleaned <- X2018.UCR.PA %>%
  rename(violent.crime = 'Violent\r\ncrime')
```

Using a backtick “`” when specifying column names helps to avoid issues with escape characters like “\r\n”. But, since there are multiple variables with wrong labels, we can change all of them simultaneously by using the pipe operator as we did in the previous chapter.

```
X2018.UCR.PA.cleaned <- X2018.UCR.PA %>%
  rename(violent.crime = 'Violent\r\ncrime') %>%
  rename(murder.manslaughter = 'Murder and \r\nnonnegligent\r\nmanslaughter') %>%
  rename(aggravated.assault = 'Aggravated\r\nassault') %>%
  rename(property.crime = 'Property\r\ncrime') %>%
  rename(larceny.theft = 'Larceny-\r\ntheft') %>%
  rename(motor.theft = 'Motor\r\nvehicle\r\ntheft')
summary(X2018.UCR.PA.cleaned)
```

You will see that all variables are properly recoded.

## Crime Rates

The term “crime rate” is commonly mentioned in the news. When we hear that crime rates are increasing, it often raises concerns, whereas hearing that crime rates are decreasing tends to make us feel relieved. Yet, what does crime rate mean? The crime rate is determined by dividing the number of Part 1 crimes by the total population and then multiplying the result by 100,000. Therefore, crime rate refers to the number of Part 1 offenses per 100,000 people in a given area. How can we create the crime rate variable for each city in our dataset?

## Mutate Function

We can use the mutate function that we learned in the previous chapter. The total number of Part 1 offenses can be computed by adding violent and property crimes to the dataset. This total number of Part 1 offenses will be divided by the population of each city, and the result will be multiplied by 100,000.

```
X2018.UCR.PA.cleaned <- X2018.UCR.PA.cleaned %>%
  mutate(crime.rate = ((violent.crime + property.crime)/
  Population)*100000)
summary(X2018.UCR.PA.cleaned)
```

The mutate() function from the dplyr package was used to add a new variable called “crime.rate” to the “X2018.UCR.PA.cleaned” dataset. Within the mutate() function, the expression (violent.crime + property.crime) / Population \* 100000 calculates the crime rate. It first sums the number of “violent.crime” and “property.crime” incidents, then divides this sum by the “Population” variable, and finally multiplies the result by 100,000 to obtain the crime rate per 100,000 people.

## Select Function

To display the names of cities and each city's crime rate next to each other in R, you can use the `select()` function from the `dplyr` package to select only these variables.

```
selected.ucr <- X2018.UCR.PA.cleaned %>%
select(City, crime.rate)
view(selected.ucr)
```

The code above created a new data frame called “selected.ucr,” containing only the “City” and “crime.rate” columns from the original data frame. Now you can see the information regarding crime rate right next to each city. For example, Abington Township, Montgomery County had 1784.976 Part 1 offenses per 100,000 residents, whereas Adamstown had 915.45504 Part 1 offenses per 100,000 residents.

## Arrange Function

Even though it is very useful to view the data frame this way, you may want to see which municipalities have higher crime rates. You can rearrange the data by using the `arrange` function.

```
arranged.data <- selected.ucr %>%
arrange(desc(crime.rate))
view(arranged.data)
```

“selected.ucr” was the data frame I used, which only contained the “City” and “crime.rate” columns. The “`arrange(desc(crime.rate))`” arranged the data frame in descending order of the “crime.rate” variable. When you view the newly arranged data frame using “`view (arranged.data)`”, you will see that Wilkes-Barre Township had the highest crime rate with 17757.009, followed by Frazer Township with 14424.779.

## Cut Function

Now, you may want to categorize towns based on population. You can create categories representing different population ranges. Cities may be divided into five different categories: small (population between 0 [inclusive] and 10,000 [exclusive]), medium (population between 10,000 [inclusive] and 50,000 [exclusive]), large (population between 50,000 [inclusive] and 100,000 [exclusive]), very large (population between 100,000 [inclusive] and 500,000 [exclusive]), and metropolitan (population 500,000 [inclusive] and above).

```
# Define the breaks for population categories breaks <- c(0, 10000, 50000, 100000,
500000, Inf)
# Define the labels for the population categories
labels <- c("Small", "Medium", "Large", "Very Large", "Metropolitan")
# Create a new variable 'population_category' based on the population ranges
X2018.UCR.PA.cleaned <- X2018.UCR.PA.cleaned %>%
mutate(population.category = cut(Population, breaks = breaks, labels = labels,
include.lowest = TRUE))
summary(X2018.UCR.PA.cleaned$population.category)
```

I defined the breaks for the population categories. Then, I defined labels for the population categories corresponding to each range. I used the `cut()` function to categorize the towns into the specified categories by population size. The resulting variable “population.category” will contain the population category for each town. Now, each town in your dataset will be categorized into one of the specified population categories based on its population size. According to the results of the summary statistics, only one metropolitan city and two very large cities exist. The majority of the cities were either small or medium.

## Group\_By Function

Now, you might want to check the average crime and the total number of crimes or each population category. We are going to use the `group_by` function from `dplyr` to perform these tasks.

```
crime.table <- X2018.UCR.PA.cleaned %>%
group_by(population.category) %>%
summarize(avg.crime.rate = mean(crime.rate, na.rm = TRUE),
total.crimes = sum(crime.rate, na.rm = TRUE))
crime.table
```

The “group\_by(population.category)” function groups the data by the “population.category” variable. Then, the “summarize()” function calculates summary statistics within each group. In this case, it computes the average crime rate (“avg.crime.rate”) and the total number of crimes (“total.crimes”) for each population category. The resulting crime.table contains summary statistics for each population category, including the average crime rate and the total number of crimes.

## Geom\_Histogram()

Finally, we can create a graph to see the distribution of crime rates across Pennsylvania cities. Specifically, we can create a histogram that is useful for displaying the distribution of continuous data by dividing it into bins (i.e., bar charts are useful for visualizing the frequency of each category in categorical variables). We are going to use ggplot2 to accomplish this task.

```
crime.rate.table <- X2018.UCR.PA.cleaned %>%
ggplot(aes(x = crime.rate, fill = ..count..)) +
geom_histogram() +
labs(x = "Crime rates", y = "Frequency", title = "Distribution of Crime Rates") +
theme_minimal()
crime.rate.table
```

“X2018.UCR.PA.cleaned” is the data frame containing the variable I wanted to visualize, and “aes(x = variable)” specifies the crime.rate variable that I wanted to plot on the x-axis. “geom\_histogram()” was the function used to create the histogram. “fill = ..count..” within the aes() function assigned a fill color to the bars based on the count of observations in each bin. This results in a gradient color scale where darker colors represent higher frequencies.

In this chapter, we reviewed how to create a variable and produce summary statistics using the concept of crime rates. In the next chapter, we will learn more about central tendency and spread, which are critical statistical measures that visualize data patterns.

---

This page titled [1.3: Creating a New Variable and Producing Summary Statistics](#) is shared under a [CC BY-SA 4.0](#) license and was authored, remixed, and/or curated by [Jaeyong Choi](#) ([The Pennsylvania Alliance for Design of Open Textbooks \(PA-ADOPT\)](#)) via [source content](#) that was edited to the style and standards of the LibreTexts platform.