The foundational success of any data analysis projects starts with proper data preparation, which includes discrete data tasks such as scraping, cleaning, transforming, munging, reshaping, among others. We can't effectively begin our data preparation tasks without first understanding our data structure, specifically whether the data we have is wide or tall.

Data structure will vary based on data collection, data storage, and intended use. For example, GIS data is often structured in a wide format so that each row represents a unique geography and each column represents a corresponding attribute. Redundant geographies might cause confusion on a map and slow the down the processing speed. Stock market transactions, however, would likely be structured in a tall format to minimize data storage size and the need for subsequent schema edits to the database.

Importantly, getting the most out of ArcGIS Insights requires using the correct data structure for the correct tool.

Wide Data

Wide data is characterized by having separate data variables in separate columns and unique measures for those variables in unique rows. Examine Table 1, it represents the population of four US States from 2015-2020 and is typical of data arranged in a wide format.

Table 1

Name	FIPS	Abbreviations	2015 Population	2016 Population	2017 Population	2018 Population	2019 Population	2020 Population
Georgia	13	GA	10,067,378	10,189,016	10,325,943	10,471,428	10,604,413	10,722,092
North Carolina	37	NC	9,932,862	10,036,881	10,151,700	10,291,929	10,455,811	10,598,314
South Carolina	45	SC	4,817,440	4,879,157	4,954,035	5,036,155	5,118,397	5,196,026
Virginia	51	VA	8,254,218	8,312,400	8,370,206	8,420,184	8,483,598	8,539,322

Source: US Census Bureau

This data set works well from a GIS perspective because each row has a unique geography with multiple corresponding attributes. When mapped in a GIS, each geography is only represented once which increases drawing speed and reduces confusion for users interacting with the map.

Tall Data

Tall data is characterized by having all data variables in a unique column and all measures for those variables in a unique column with each row representing a unique observation. Examine Table 2, it represents the monthly closing price of four stock indexes over the past 6 months and is typical of data arranged in a tall format.

Table 2

Name	Month	Closing Price ($)
S&P 500	May	4,298
S&P 500	June	4,395
S&P 500	July	4,523
S&P 500	August	4,308
S&P 500	September	4,605
S&P 500	October	4,691
Dow Jones Industrial Average	May	34,503
Dow Jones Industrial Average	June	34,935
Dow Jones Industrial Average	July	35,361
Dow Jones Industrial Average	August	33,844
Dow Jones Industrial Average	September	35,820
Dow Jones Industrial Average	October	35,814
NASDAQ Composite	May	14,504
NASDAQ Composite	June	14,673
NASDAQ Composite	July	15,259
NASDAQ Composite	August	14,449
NASDAQ Composite	September	15,498
NASDAQ Composite	October	15,775
Russell 2000	May	2,311
Russell 2000	June	2,226

Russell 2000	July	2,274
Russell 2000	August	2,204
Russell 2000	September	2,297
Russell 2000	October	2,328

Source: Yahoo Finance

This data set works well for financial analysts as it's highly readable by financial modeling tools and could scale easily without changing the schema of the database.

It's important to note that data structures exist on a spectrum, they can be tall, wide, or some variation of tall and wide. Using the ArcGIS Insights scripting console, we're able to reshape the structure of our data.

Reshaping Data (Wide to Tall)

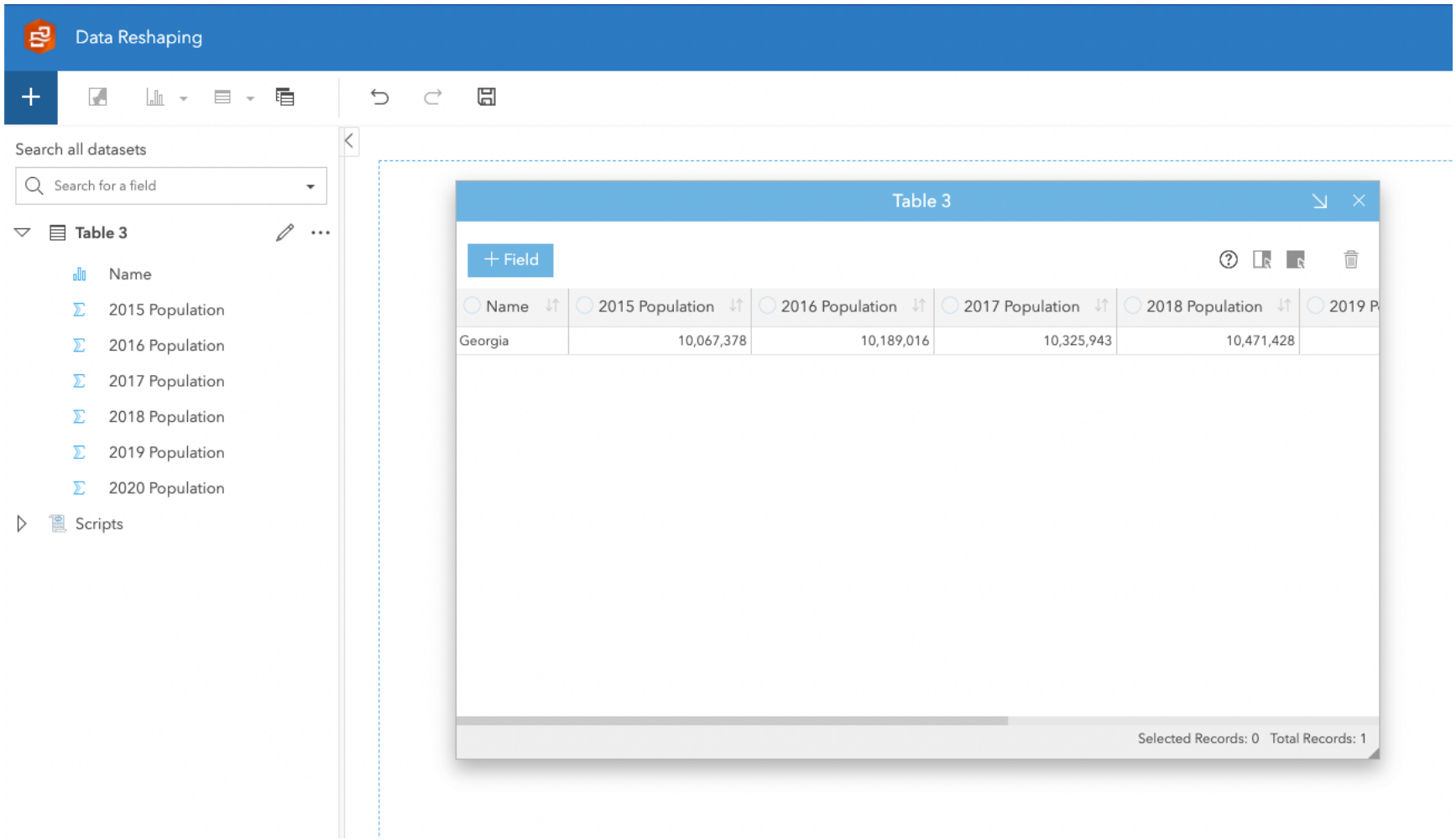
There are many terms used to describe the reshaping of data, but we will use pivot (wide to tall) and unpivot (tall to wide) to describe the transformation between the two formats. Examine Table 3, it's a bit simpler than our first table but still represents the population of four US states from 2015-2020 and is arranged in a wide format.

Table 3

Name	2015 Population	2016 Population	2017 Population	2018 Population	2019 Population	2020 Population
Georgia	10,067,378	10,189,016	10,325,943	10,471,428	10,604,413	10,722,092
North Carolina	9,932,862	10,036,881	10,151,700	10,291,929	10,455,811	10,598,314
South Carolina	4,817,440	4,879,157	4,954,035	5,036,155	5,118,397	5,196,026
Virginia	8,254,218	8,312,400	8,370,206	8,420,184	8,483,598	8,539,322

Source: US Census Bureau

Importing this into Insights will give us the following setup.



While we certainly can map this data easily, our data visualization options are limited by the structure of the data. To increase our options, we'll need to reshape our data from wide to tall.

For help getting started with Scripting in Insights, see here

The ArcGIS Insights scripting console allows you to achieve advanced data manipulation and analysis using either Python or R and both are detailed here. Import the appropriate script and review the directions.

DataReshapeTemplate_WidetoTall_Python.ipynb

Python 3 (ip...) Select all | Clear selection

In[]
```
# Select fields and drag/drop from Insights data pane

data1 = Table 3 (7 Fields)
```

In[]
```
# Display list of column names

data1.columns
```

In[]
```
# Remove string from year variables

data1.rename(columns={'f2015_population':'2015',
                      'f2016_population':'2016',
                      'f2017_population':'2017',
                      'f2018_population':'2018',
                      'f2019_population':'2019',
                      'f2020_population':'2020'},
             inplace=True)
```

In[]
```
# Use .melt() to pivot the data

data2 = data1.melt(id_vars = ['name'],
                   var_name = 'Year',
                   value_name = 'Population')
```

In[]
```
# Display list of column names

data2.columns
```

In[]
```
# Rename columns (if necessary)

data2.rename(columns={'name':'Name'},
             inplace=True)
```

In[]
```
# Add data to Insights

%insights_return(data2)
```

DataReshapeTemplate_WidetoTall_R.ipynb

R

Select all | Clear selection

In[]
```r
# Select fields and drag/drop from Insights data pane

data1 = Table 3 (7 Fields)
```

In[]
```r
# Display list of column names

colnames(data1)
```

In[]
```r
# Remove string from year variables

names(data1)[which(names(data1) %in% c("f2015_population",
"f2016_population", "f2017_population", "f2018_population",
"f2019_population", "f2020_population"))] <- c("2015", "2016", "2017",
"2018", "2019", "2020")
```

In[]
```r
# Use melt() to pivot the data

data2 <- melt(data = data1,
              id.vars = c("name"),
              variable.name = "Year",
              value.name = "Population")
```

In[]
```r
# Display list of column names

colnames(data2)
```

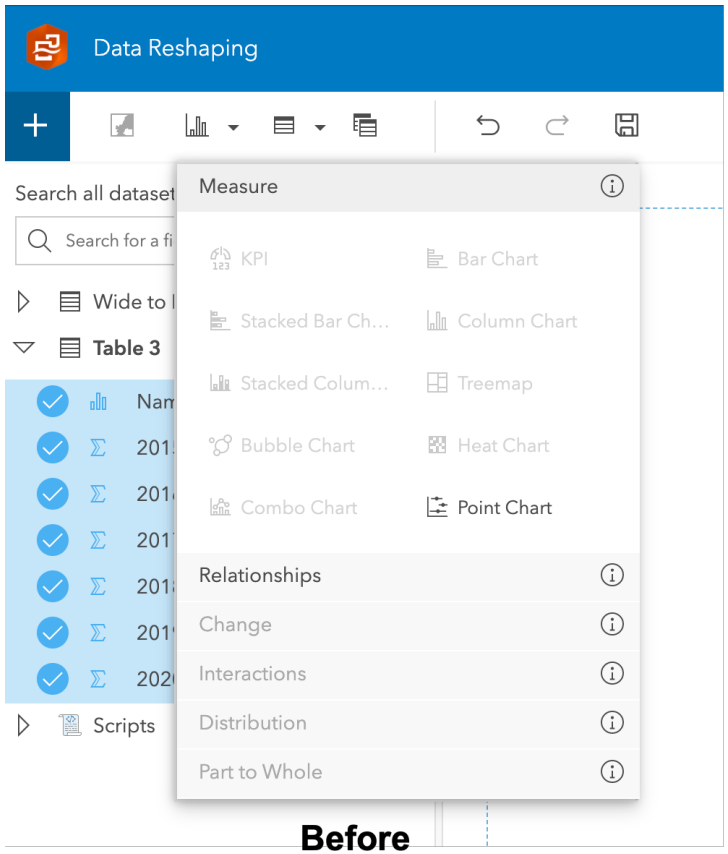
In[]
```r
# Rename columns (if necessary)

names(data2)[which(names(data2) %in% c("name"))] <- c("Name")
```

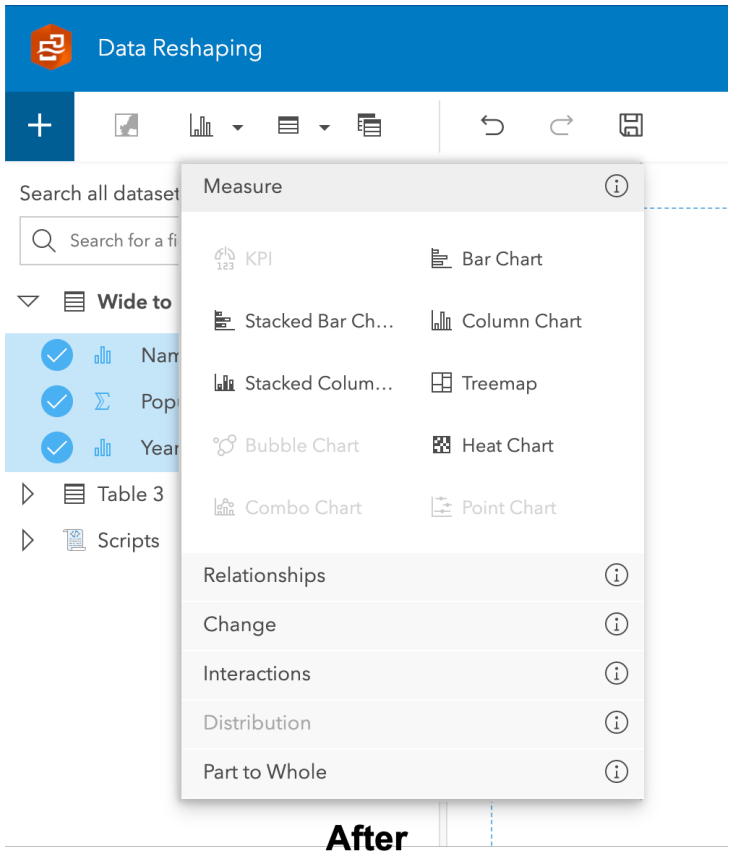
In[]
```r
# Add data to Insights

%insights_return(data2)
```

After running the appropriate script, our data has been transformed from wide to tall and we have access to a greater number of data visualizations.



Before



After

Our data will likely be more complex than this however and may include multiple variables (columns) that we don't want to pivot. Let's look again at Table 1, it represents the population of four US states from 2015-2020 but in addition to {Name} it includes {FIPS} and {Abbreviations} as identifier variables.

Table 1

Name	FIPS	Abbreviations	2015 Population	2016 Population	2017 Population	2018 Population	2019 Population	2020 Population
Georgia	13	GA	10,067,378	10,189,016	10,325,943	10,471,428	10,604,413	10,722,092
North Carolina	37	NC	9,932,862	10,036,881	10,151,700	10,291,929	10,455,811	10,598,314
South Carolina	45	SC	4,817,440	4,879,157	4,954,035	5,036,155	5,118,397	5,196,026
Virginia	51	VA	8,254,218	8,312,400	8,370,206	8,420,184	8,483,598	8,539,322

Source: US Census Bureau

We want to reshape Table 1 but this time we want to keep three fields ({Name), {FIPS}, and {Abbreviations}) as our identifier variables. To accomplish this, we'll use the **id_vars** (Python) and **id.vars** (R) parameters within our melt functions.

Python

In[]
```
# Basic (1 ID Variable)

data2 = data1.melt(id_vars = ['Name'],
                   var_name = 'Year',
                   value_name = 'Population')
```

In[]
```
# Multi (2 or more ID variables)

data2 = data1.melt(id_vars = ['Name', 'FIPS', 'Abbreviations'],
                   var_name = 'Year',
                   value_name = 'Population')
```

R

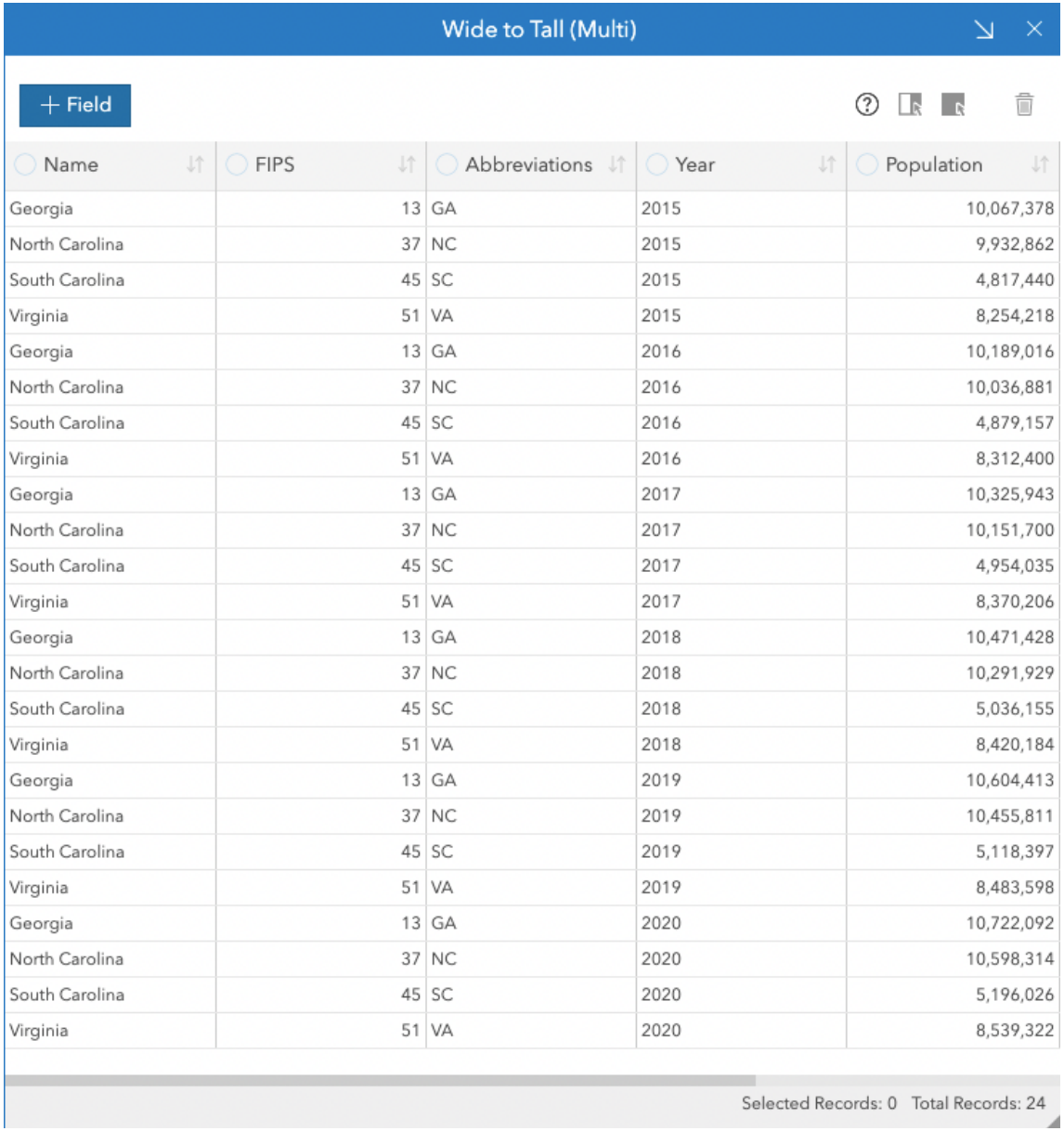
In[]
```
# Basic (1 ID Variable)

data2 <- melt(data = data1,
              id.vars = c("Name"),
              variable.name = "Year",
              value.name = "Population")
```

In[]
```
# Multi (2 or more ID variables)

data2 <- melt(data = data1,
              id.vars = c("Name", "FIPS", "Abbreviations"),
              variable.name = "Year",
              value.name = "Population")
```

Our data has again been transformed from Wide to Tall but this time we've retained our identifier variables.


Wide to Tall (Multi)
+ Field
Name
FIPS
Abbreviations
Year
Population
Georgia 13 GA 2015 10,067,378
North Carolina 37 NC 2015 9,932,862
South Carolina 45 SC 2015 4,817,440
Virginia 51 VA 2015 8,254,218
Georgia 13 GA 2016 10,189,016
North Carolina 37 NC 2016 10,036,881
South Carolina 45 SC 2016 4,879,157
Virginia 51 VA 2016 8,312,400
Georgia 13 GA 2017 10,325,943
North Carolina 37 NC 2017 10,151,700
South Carolina 45 SC 2017 4,954,035
Virginia 51 VA 2017 8,370,206
Georgia 13 GA 2018 10,471,428
North Carolina 37 NC 2018 10,291,929
South Carolina 45 SC 2018 5,036,155
Virginia 51 VA 2018 8,420,184
Georgia 13 GA 2019 10,604,413
North Carolina 37 NC 2019 10,455,811
South Carolina 45 SC 2019 5,118,397
Virginia 51 VA 2019 8,483,598
Georgia 13 GA 2020 10,722,092
North Carolina 37 NC 2020 10,598,314
South Carolina 45 SC 2020 5,196,026
Virginia 51 VA 2020 8,539,322
Selected Records: 0 Total Records: 24

Reshaping Data (Tall to Wide)

The process for unpivoting (tall to wide) the data is very similar but let's first examine why it might be needed.

Table 4 is representing the same data as Table 3 but is arranged in a tall format.

Table 4

Name	Year	Population
Georgia	2015	10,067,378
Georgia	2016	10,189,016
Georgia	2017	10,325,943
Georgia	2018	10,471,428
Georgia	2019	10,604,413
Georgia	2020	10,722,092
North Carolina	2015	9,932,862
North Carolina	2016	10,036,881
North Carolina	2017	10,151,700

North Carolina	2018	10,291,929
North Carolina	2019	10,455,811
North Carolina	2020	10,598,314
South Carolina	2015	4,817,440
South Carolina	2016	4,879,157
South Carolina	2017	4,954,035
South Carolina	2018	5,036,155
South Carolina	2019	5,118,397
South Carolina	2020	5,196,026
Virginia	2015	8,254,218
Virginia	2016	8,312,400
Virginia	2017	8,370,206
Virginia	2018	8,420,184
Virginia	2019	8,483,598
Virginia	2020	8,539,322

Source: US Census Bureau

Let's say we want to present the data in a way that easily allows our viewers to compare population growth over time.

Again, we'll use the ArcGIS Insights scripting console to reshape our data.

DataReshapeTemplate_TalltoWide_Python.ipynb

Python 3 (ip...

Select all | Clear selection

In[]
```
# Select fields and drag/drop from Insights data pane

data1 = Table 4 (3 Fields)
```

In[]
```
# Display list of column names

data1.columns
```

In[]
```
# Use pd.pivot_table() to unpivot the data

data2 = data1.pivot_table(index = ['name'],
                          columns = 'year',
                          values = 'population').reset_index()
```

In[]
```
# Display list of column names

data2.columns
```

In[]
```
# Rename columns (if necessary)

data2.rename(columns={'name':'Name',
                      '2015':'Population2015',
                      '2016':'Population2016',
                      '2017':'Population2017',
                      '2018':'Population2018',
                      '2019':'Population2019',
                      '2020':'Population2020'},
             inplace=True)
```

In[]
```
# Add data to Insights

%insights_return(data2)
```

DataReshapeTemplate_TalltoWide_R.ipynb

R

Select all | Clear selection

In[]
```r
# Select fields and drag/drop from Insights data pane

data1 = Table 4 (3 Fields)
```

In[]
```r
# Display list of column names

colnames(data1)
```

In[]
```r
# Use pd.pivot_table() to unpivot the data

data2 <- reshape(data = data1,
                 idvar = c("name"),
                 timevar = "year",
                 v.names = "population",
                 direction = "wide")
```

In[]
```r
# Display list of column names

colnames(data2)
```

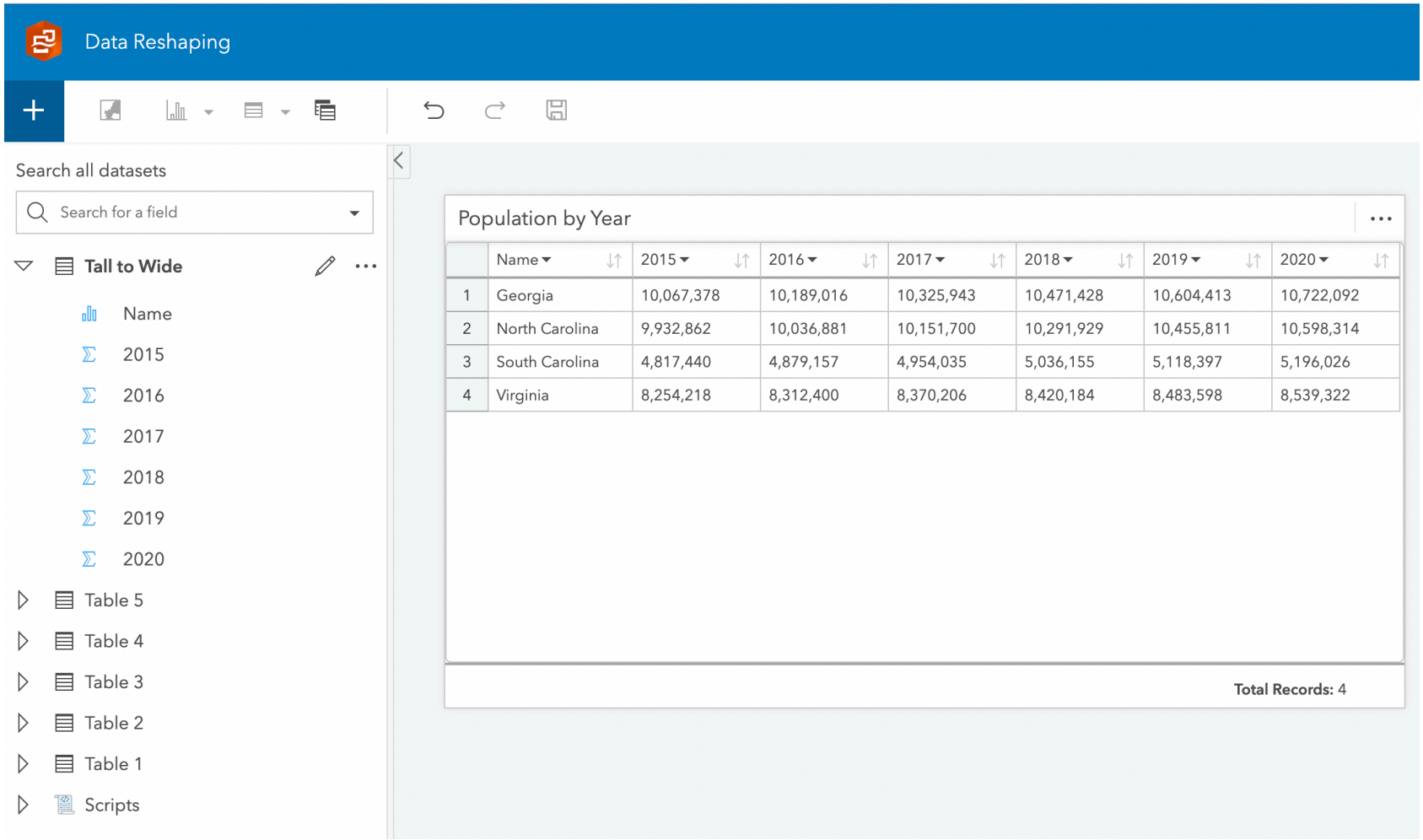
In[]
```r
# Rename columns (if necessary)

names(data2)[which(names(data2) %in% c("name", "population.2015",
"population.2016", "population.2017", "population.2018",
"population.2019", "population.2020"))] <- c("Name", "Population2015",
"Population2016", "Population2017", "Population2018",
"Population2019", "Population2020")
```

In[]
```r
# Add data to Insights

%insights_return(data2)
```

After running the script, our data is transformed from tall to wide and we now have the ability display our data sequentially by year using the reference table tool.



	Name	2015	2016	2017	2018	2019	2020
1	Georgia	10,067,378	10,189,016	10,325,943	10,471,428	10,604,413	10,722,092
2	North Carolina	9,932,862	10,036,881	10,151,700	10,291,929	10,455,811	10,598,314
3	South Carolina	4,817,440	4,879,157	4,954,035	5,036,155	5,118,397	5,196,026
4	Virginia	8,254,218	8,312,400	8,370,206	8,420,184	8,483,598	8,539,322

Again, if we have columns we don't want reshaped, we can handle that as well. In Table 5, we don't want {Name}, {FIPS}, or {Abbreviations} transformed.

Table 5

Name	FIPS	Abbreviations	Year	Population
Georgia	13	GA	2015	10,067,378
Georgia	13	GA	2016	10,189,016
Georgia	13	GA	2017	10,325,943
Georgia	13	GA	2018	10,471,428
Georgia	13	GA	2019	10,604,413
Georgia	13	GA	2020	10,722,092
North Carolina	37	NC	2015	9,932,862
North Carolina	37	NC	2016	10,036,881
North Carolina	37	NC	2017	10,151,700
North Carolina	37	NC	2018	10,291,929
North Carolina	37	NC	2019	10,455,811
North Carolina	37	NC	2020	10,598,314
South Carolina	45	SC	2015	4,817,440
South Carolina	45	SC	2016	4,879,157
South Carolina	45	SC	2017	4,954,035
South Carolina	45	SC	2018	5,036,155

South Carolina	45	SC	2019	5,118,397
South Carolina	45	SC	2020	5,196,026
Virginia	51	VA	2015	8,254,218
Virginia	51	VA	2016	8,312,400
Virginia	51	VA	2017	8,370,206
Virginia	51	VA	2018	8,420,184
Virginia	51	VA	2019	8,483,598
Virginia	51	VA	2020	8,539,322

Source: US Census Bureau

To accomplish this, we'll use the **index** (Python) and **idvar** (R) parameters.

Python

In[]
```
# Basic (1 ID Variable)

data2 = data1.pivot_table(index = ['Name'],
                          columns = 'Year',
                          values = 'Population').reset_index()
```

In[]
```
# Multi (2 or more ID variables)

data2 = data1.pivot_table(index = ['Name', 'FIPS', 'Abbreviations'],
                          columns = 'Year',
                          values = 'Population').reset_index()
```

R

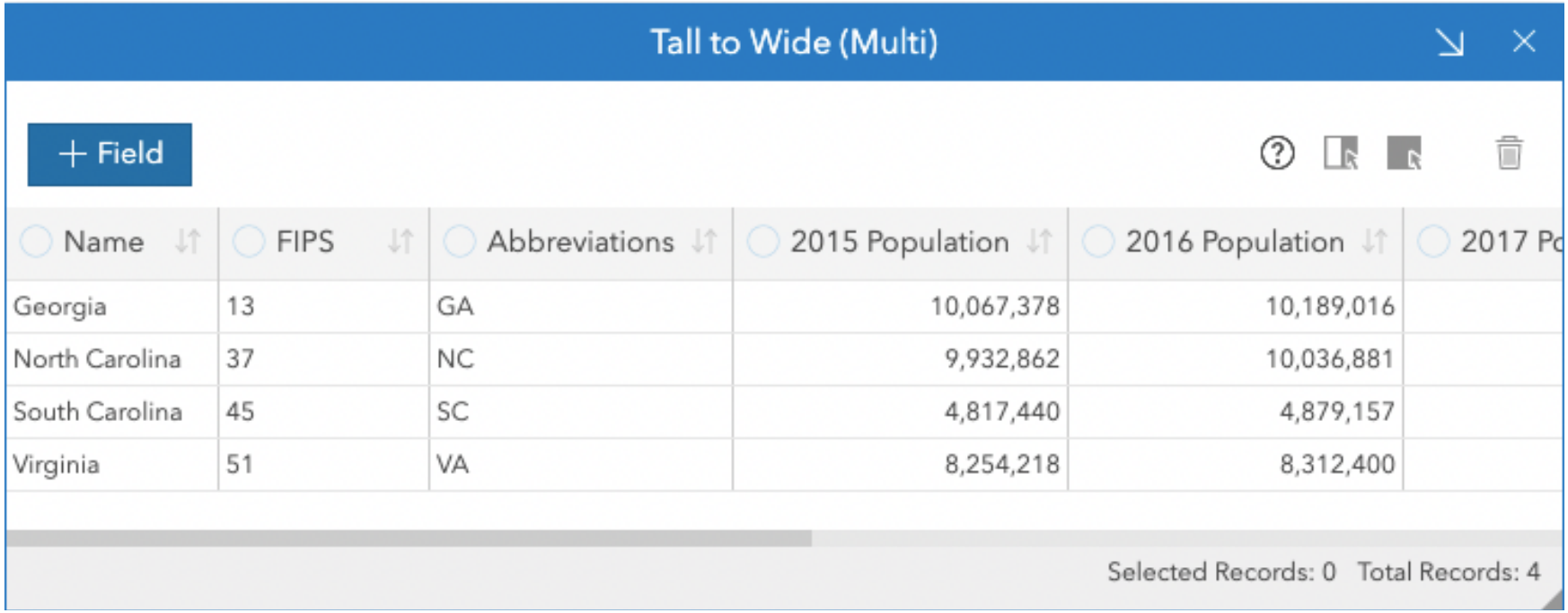
In[]
```r
# Basic (1 ID Variable)

data2 <- reshape(data = data1,
                 idvar = c("Name"),
                 timevar = "Year",
                 v.names = "Population",
                 direction = "wide")
```

In[]
```r
# Multi (2 or more ID variables)

data2 <- reshape(data = data1,
                 idvar = c("Name", "FIPS", "Abbreviations"),
                 timevar = "Year",
                 v.names = "Population",
                 direction = "wide")
```

Reshaping complete and identifier variables remain unpivoted.



And that's it! Data reshaping from wide to tall and back again.

ArcGIS Insights is a powerful tool for location analytics and proper data structure unlocks its full potential. Hopefully, these tools will help you ask new questions of your data, unlock new patterns, and provide insight to your organization.