Datasets

The **gt** package is equipped with six datasets. Use them to experiment with the package functions, they come in all shapes and sizes. Many examples in the internal help documents use these datasets to quickly demonstrate key features of **gt**.

`countrypops` : Yearly populations of countries from 1960 to 2017

A dataset that presents yearly, total populations of countries. Total population is based on counts of all residents regardless of legal status or citizenship. Country identifiers include the English-language country names, and the 2- and 3-letter ISO 3166-1 country codes. Each row contains a population value for a given year (from 1960 to 2017). Any `NA` values for populations indicate the non-existence of the country during that year.

EXAMPLE

Here is a glimpse at the data available in `countrypops` .

```
dplyr::glimpse(countrypops)
```

```
## Rows: 12,470
## Columns: 5
## $ country_name   <chr> "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", "Aruba", …
## $ country_code_2 <chr> "AW", "AW", "AW", "AW", "AW", "AW", "AW", "AW", "AW", …
## $ country_code_3 <chr> "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "ABW", "ABW"…
## $ year           <int> 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, …
## $ population     <int> 54211, 55438, 56225, 56695, 57032, 57360, 57715, 58055…
```

`sza` : Twice hourly solar zenith angles by month & latitude

This dataset contains solar zenith angles (in degrees, with the range of 0-90) every half hour from 04:00 to 12:00, true solar time. This set of values is calculated on the first of every month for 4 different northern hemisphere latitudes. For determination of afternoon values, the presented tabulated values are symmetric about noon.

EXAMPLE

Here is a glimpse at the data available in `sza` .

```
dplyr::glimpse(sza)
```

```
## Rows: 816
## Columns: 4
## $ latitude <dbl> 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, …
## $ month    <fct> jan, jan, jan, jan, jan, jan, jan, jan, jan, jan, jan, jan, …
## $ tst      <chr> "0400", "0430", "0500", "0530", "0600", "0630", "0700", "073…
## $ sza      <dbl> NA, NA, NA, NA, NA, NA, 84.9, 78.7, 72.7, 66.1, 61.5, 56.5, …
```

`gtcars` : Deluxe automobiles from the 2014-2017 period

Expensive and fast cars. Not your father's `mtcars`. Each row describes a car of a certain make, model, year, and trim. Basic specifications such as horsepower, torque, EPA MPG ratings, type of drivetrain, and transmission characteristics are provided. The country of origin for the car manufacturer is also given.

EXAMPLE

Here is a glimpse at the data available in `gtcars`.

```
dplyr::glimpse(gtcars)
```

```
## Rows: 47
## Columns: 15
## $ mfr         <chr> "Ford", "Ferrari", "Ferrari", "Ferrari", "Ferrari", "Ferr…
## $ model       <chr> "GT", "458 Speciale", "458 Spider", "458 Italia", "488 GT…
## $ year        <dbl> 2017, 2015, 2015, 2014, 2016, 2015, 2017, 2015, 2015, 201…
## $ trim        <chr> "Base Coupe", "Base Coupe", "Base", "Base Coupe", "Base C…
## $ bdy_style   <chr> "coupe", "coupe", "convertible", "coupe", "coupe", "conve…
## $ hp          <dbl> 647, 597, 562, 562, 661, 553, 680, 652, 731, 949, 573, 54…
## $ hp_rpm      <dbl> 6250, 9000, 9000, 9000, 8000, 7500, 8250, 8000, 8250, 900…
## $ trq         <dbl> 550, 398, 398, 398, 561, 557, 514, 504, 509, 664, 476, 43…
## $ trq_rpm     <dbl> 5900, 6000, 6000, 6000, 3000, 4750, 5750, 6000, 6000, 675…
## $ mpg_c       <dbl> 11, 13, 13, 13, 15, 16, 12, 11, 11, 12, 21, 16, 11, 16, 1…
## $ mpg_h       <dbl> 18, 17, 17, 17, 22, 23, 17, 16, 16, 16, 22, 22, 18, 20, 2…
## $ drivetrain  <chr> "rwd", "rwd", "rwd", "rwd", "rwd", "rwd", "awd", "awd", "…
## $ trsmn       <chr> "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a", "7a…
## $ ctry_origin <chr> "United States", "Italy", "Italy", "Italy", "Italy", "Ita…
## $ msrp        <dbl> 447000, 291744, 263553, 233509, 245400, 198973, 298000, 2…
```

`sp500`: Daily S&P 500 Index data from 1950 to 2015

This dataset provides daily price indicators for the S&P 500 index from the beginning of 1950 to the end of 2015. The index includes 500 leading companies and captures about 80% coverage of available market capitalization. This is admittedly a **very boring dataset**, and I guess we're `OK` with that.

EXAMPLE

Here is a glimpse at the data available in `sp500`.

```
dplyr::glimpse(sp500)
```

```
## Rows: 16,607
## Columns: 7
## $ date      <date> 2015-12-31, 2015-12-30, 2015-12-29, 2015-12-28, 2015-12-24…
## $ open      <dbl> 2060.59, 2077.34, 2060.54, 2057.77, 2063.52, 2042.20, 2023.…
## $ high      <dbl> 2062.54, 2077.34, 2081.56, 2057.77, 2067.36, 2064.73, 2042.…
## $ low       <dbl> 2043.62, 2061.97, 2060.54, 2044.20, 2058.73, 2042.20, 2020.…
## $ close     <dbl> 2043.94, 2063.36, 2078.36, 2056.50, 2060.99, 2064.29, 2038.…
## $ volume    <dbl> 2655330000, 2367430000, 2542000000, 2492510000, 1411860000,…
## $ adj_close <dbl> 2043.94, 2063.36, 2078.36, 2056.50, 2060.99, 2064.29, 2038.…
```

`pizzaplace`: A year of pizza sales from a pizza place

A synthetic dataset that describes pizza sales for a pizza place somewhere in the US. While the contents are artificial, the ingredients used to make the pizzas are far from it. There are 32 different pizzas that fall into 4 different categories: `classic` (classic pizzas: 'You probably had one like it before, but never like this!'), `chicken` (pizzas with chicken as a major ingredient: 'Try the Southwest Chicken Pizza! You'll love it!'), `supreme` (pizzas that try a little harder: 'My Soppressata pizza uses only the finest salami from my personal salumist!'), and, `veggie` (pizzas without any meats whatsoever: 'My Five Cheese pizza has so many cheeses, I can only offer it in Large Size!').

EXAMPLE

Here is some of the data somehow made available by the `pizzaplace`.

```
dplyr::glimpse(pizzaplace)
```

```
## Rows: 49,574
## Columns: 7
## $ id    <chr> "2015-000001", "2015-000002", "2015-000002", "2015-000002", "20…
## $ date  <chr> "2015-01-01", "2015-01-01", "2015-01-01", "2015-01-01", "2015-0…
## $ time  <chr> "11:38:36", "11:57:40", "11:57:40", "11:57:40", "11:57:40", "11…
## $ name  <chr> "hawaiian", "classic_dlx", "mexicana", "thai_ckn", "five_cheese…
## $ size  <chr> "M", "M", "M", "L", "L", "L", "L", "M", "M", "M", "S", "S", "S"…
## $ type  <chr> "classic", "classic", "veggie", "chicken", "veggie", "supreme",…
## $ price <dbl> 13.25, 16.00, 16.00, 20.75, 18.50, 20.75, 20.75, 16.50, 16.50, …
```

`exibble`: A toy example tibble for testing with **gt**: `exibble`

This tibble contains data of a few different classes, which makes it well-suited for quick experimentation with the functions in this package. It contains only eight rows with numeric, character, and factor columns. The last 4 rows contain `NA` values in the majority of this tibble's columns (1 missing value per column). The `date`, `time`, and `datetime` columns are character-based dates/times in the familiar ISO 8601 format. The row and group columns provide for unique rownames and two groups (`grp_a` and `grp_b`) for experimenting with the `gt()` function's `rowname_col` and `groupname_col` arguments.

EXAMPLE

Here is the `exibble`:

```
exibble
```

```
## # A tibble: 8 x 9
##           num char      fctr  date     time  datetime       currency row   group
##         <dbl> <chr>     <fct> <chr>    <chr> <chr>             <dbl> <chr> <chr>
## 1       0.111 apricot   one   2015-01… 13:35 2018-01-01 02…    50.0  row_1 grp_a
## 2       2.22  banana    two   2015-02… 14:40 2018-02-02 14…    18.0  row_2 grp_a
## 3      33.3   coconut   three 2015-03… 15:45 2018-03-03 03…     1.39 row_3 grp_a
## 4     444.    durian    four  2015-04… 16:50 2018-04-04 15… 65100    row_4 grp_a
## 5    5550     <NA>      five  2015-05… 17:55 2018-05-05 04…  1326.   row_5 grp_b
## 6      NA     fig       six   2015-06… <NA>  2018-06-06 16…    13.3  row_6 grp_b
## 7  777000     grapefru… seven <NA>     19:10 2018-07-07 05…    NA    row_7 grp_b
## 8 8880000     honeydew  eight 2015-08… 20:20 <NA>               0.44 row_8 grp_b
```
