Developing in-house software

Why I did it & you should too

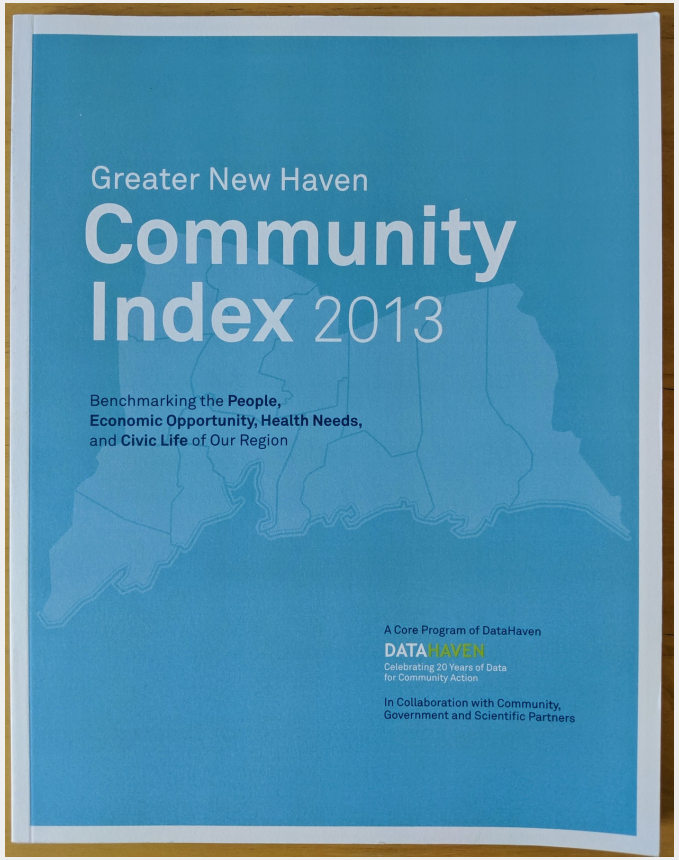
Camille Seaberry

DataHaven

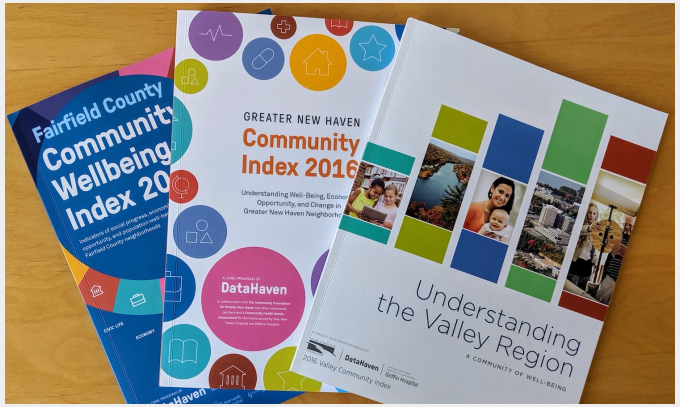
Follow along: ct-data-haven.github.io/datadev

DataHaven's Community Index

2013



2016



2019: the takeover

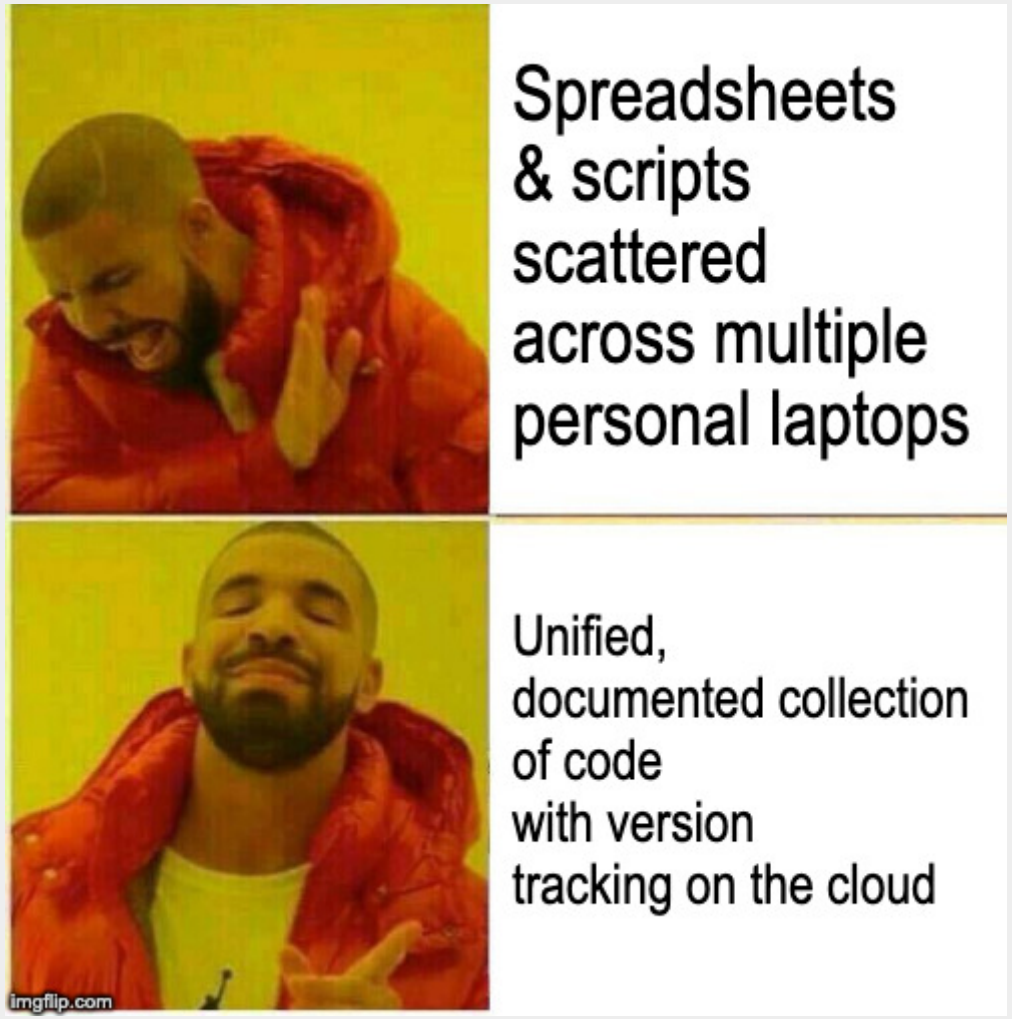
Community wellbeing comes from a number of different factors

Components of the DataHaven Community Index, Greater New Haven, 2019

		Greater New Haven	CT	US	New Haven	Best	Worst
Index components	Life expectancy	N/A	80 yrs	79 yrs	78 yrs	CT 80 yrs	West Haven 78 yrs
	Health insurance	94%	94%	90%	90%	Outer Ring 97%	New Haven, lower-income 88%
	Opportunity youth	4%	5%	7%	7%	West Haven 2%	New Haven, lower-income 9%
	High school graduates	91%	90%	87%	85%	Outer Ring 95%	New Haven, lower-income 79%
	Workers with short commute	70%	65%	63%	74%	New Haven, other 78%	US 63%
	Preschool enrollment	64%	64%	48%	61%	Outer Ring 71%	East Haven 37%
	Poverty	12%	10%	15%	26%	Outer Ring 4%	New Haven, lower-income 31%
	Young child poverty	18%	15%	22%	34%	Hamden 5%	New Haven, lower-income 44%
	Unemployment	7%	7%	7%	10%	Outer Ring 5%	New Haven, lower-income 14%
	Young adults in labor force	25%	24%	26%	32%	New Haven, other 34%	Outer Ring 21%
	Severe housing cost burden	20%	17%	15%	30%	Outer Ring 14%	New Haven, lower-income 36%
	Median household income	$75k	$74k	$58k	$39k	Outer Ring $94k	New Haven, lower-income $34k
Overall (0 - 1000)	Community index	629	621	551	468	Outer Ring 728	New Haven, lower-income 365

Relative rank: Better, Average, Worse, Not avail.

Spreadsheet sprawl



Installing packages is easy

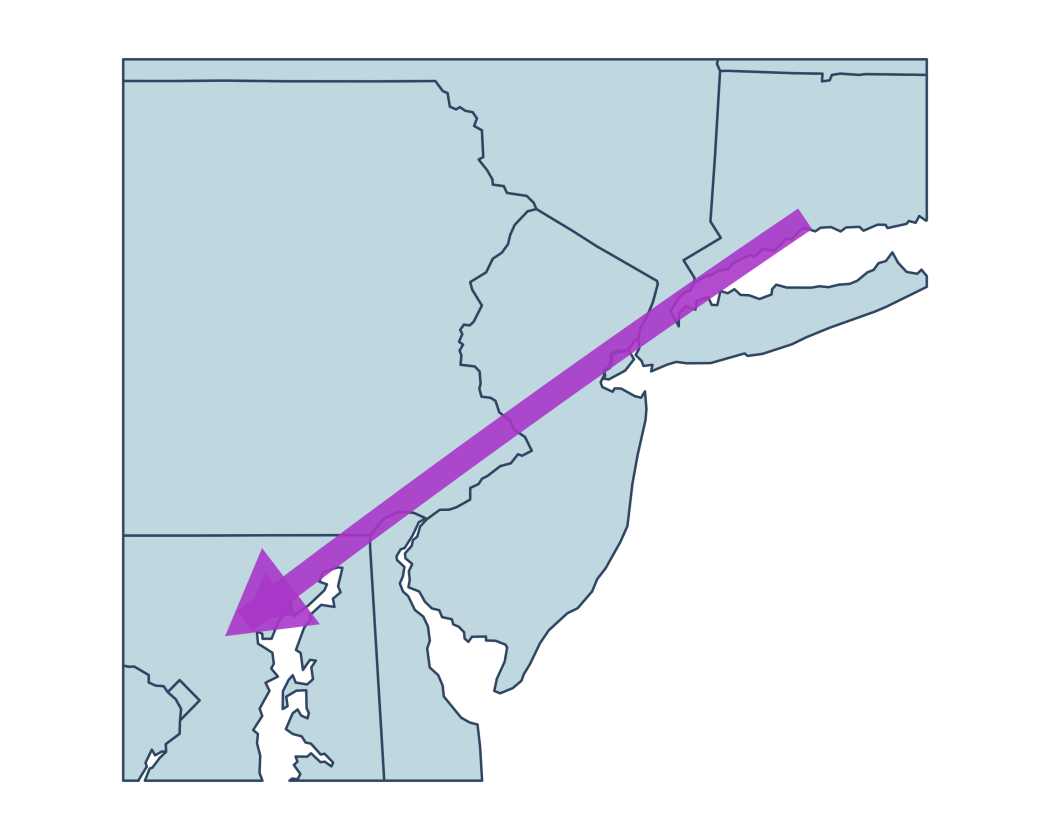
```
install_github("camille-s/camiller")
install_github("CT-Data-Haven/cwi")
```

Created by Gan Khoon Lay
from Noun Project

Shifting my thinking: toward sustainable & reproducible work

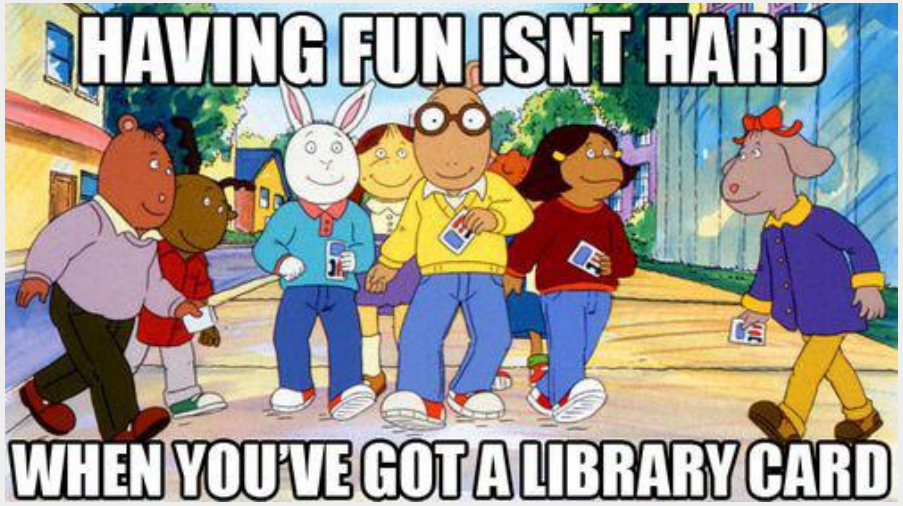
Unhappy Mother's Day, but a new appreciation of behind-the-scenes information

Shifting my thinking: toward sustainable & reproducible work



I moved 300 miles away to Baltimore

What's a library?




```r
library(tidyverse)    # sets up LOTS of functions, how I start my mornings
library(tidycensus)   # fetches data from Census API
library(camiller)     # first in-house library
library(cwi)          # second in-house library
library(showtext)     # use nice fonts in plots
library(sf)           # work with geospatial data & make maps
library(patchwork)    # layout plots together
library(lubridate)    # parse dates
```

Functions! If only...

```r
leave_the_house <- function(date = today(), biking = TRUE, working = TRUE) {
  day_of_week <- wday(date, label = TRUE, abbr = FALSE)
  always_need <- c("keys", "phone", "wallet", "meds")
  sometimes_need <- c()
  if (biking) {
    sometimes_need <- c(sometimes_need, "helmet")
  } else {
    sometimes_need <- c(sometimes_need, "bus card")
  }
  if (working) {
    sometimes_need <- c(sometimes_need, "laptop")
  }
  need <- c(always_need, sometimes_need)

  cat(
    sprintf("Happy %s! Today you need:", day_of_week), "\n",
    paste(need, collapse = ", ")
  )
}
```

Functions! If only...

```
leave_the_house(biking = TRUE, working = FALSE)
```

```
Happy Saturday! Today you need:
 keys, phone, wallet, meds, helmet
```

Functions: reduce repetition & clutter

Tedious and messy

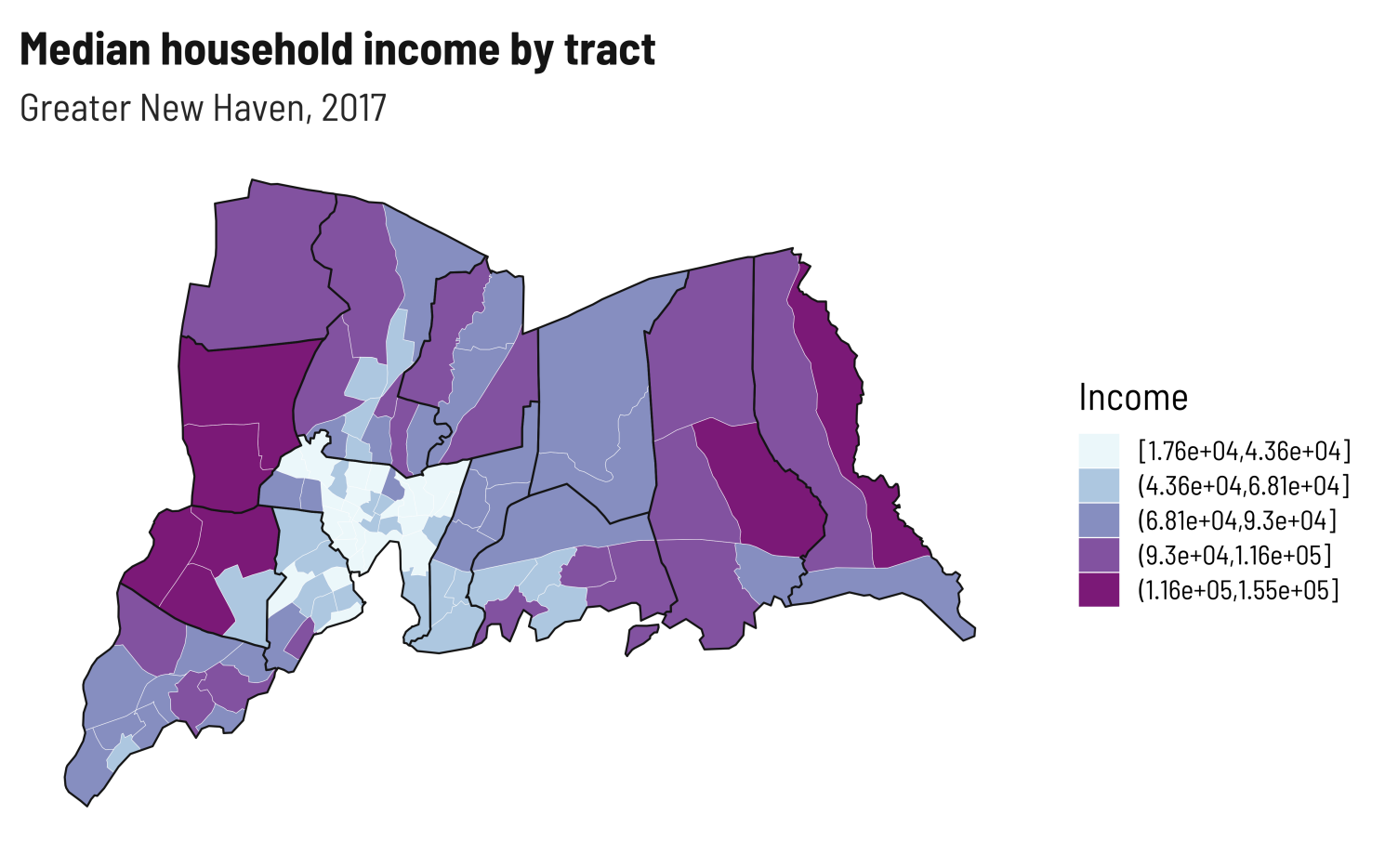
```
income_us ← get_acs("us", table = "B19013", year = 2017)
income_state ← get_acs("state", table = "B19013", year = 2017)
income_msa ← get_acs("metropolitan statistical area/micropolitan statistical area", table
income_county ← get_acs("county", table = "B19013", state = "09", year = 2017)
income_towns ← get_acs("county subdivision", table = "B19013", state = "09", year = 2017)
income ← bind_rows(income_us, income_state, income_msa, income_county, income_towns)

# get rid of those extra tables
rm(income_us, income_state, income_msa, income_county, income_towns)
```

Nice n clean

```
income ← multi_geo_acs(table = "B19013", year = 2017, us = TRUE, msa = TRUE)
```

Functions: I *swear* I did this last week!



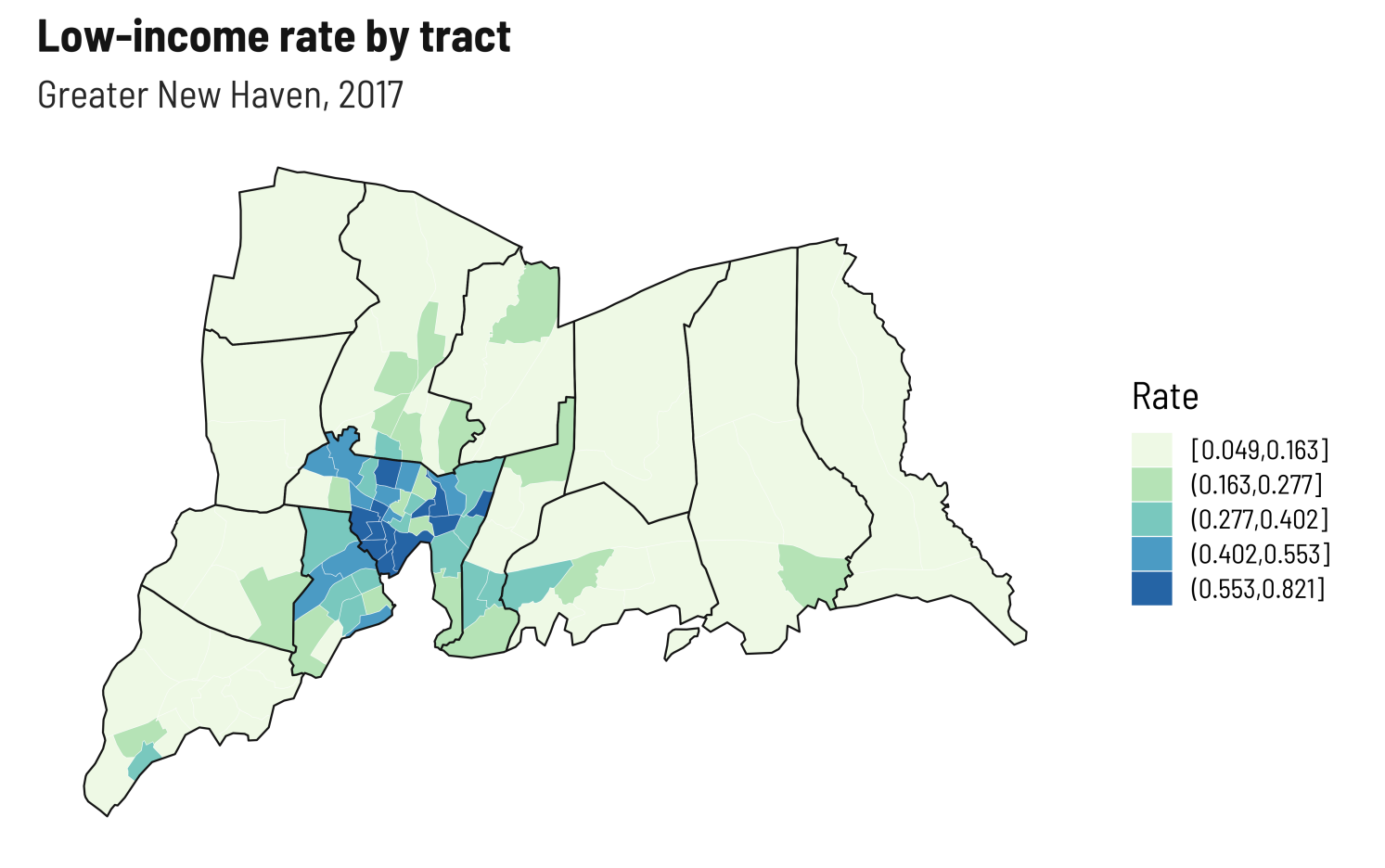
:(

- [1.76e+04,4.36e+04]
- (4.36e+04,6.81e+04]
- (6.81e+04,9.3e+04]
- (9.3e+04,1.16e+05]
- (1.16e+05,1.55e+05]

:)

- $18k to $44k
- $44k to $68k
- $68k to $93k
- $93k to $116k
- $116k to $155k

Functions: make it scale



:(

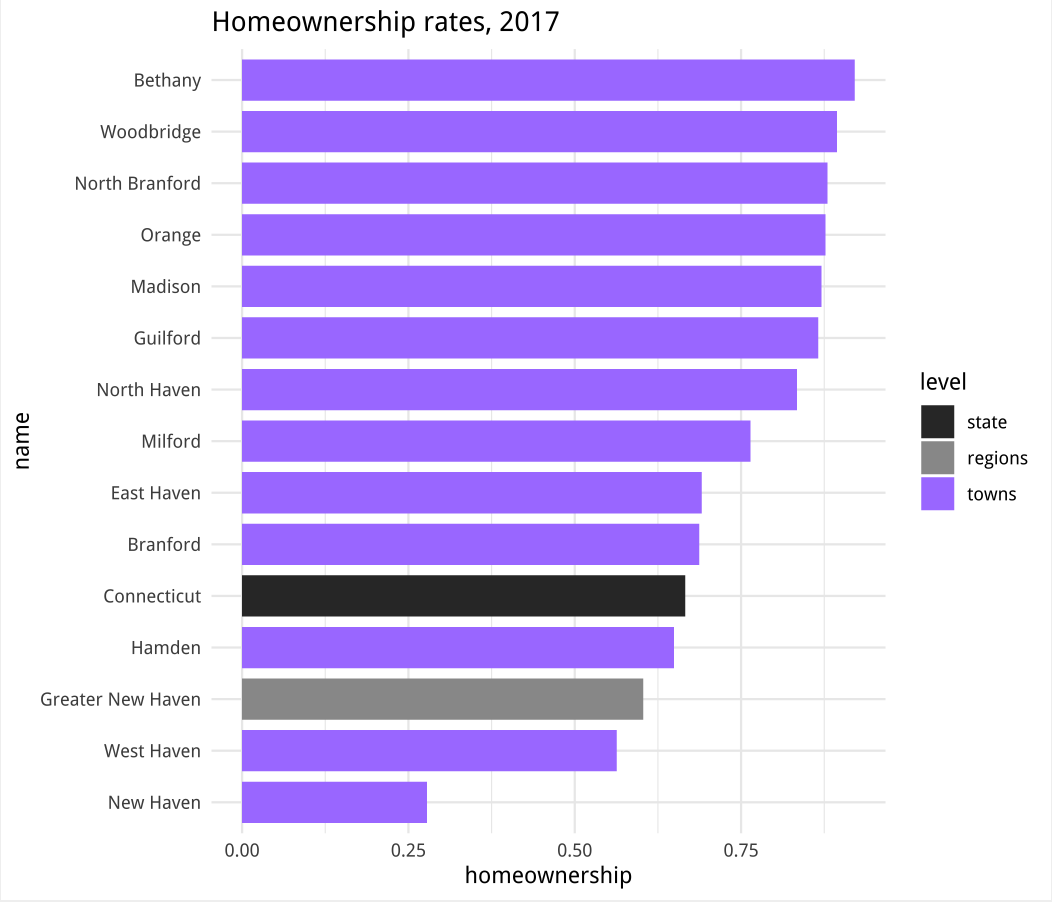
- [0.049,0.163]
- (0.163,0.277]
- (0.277,0.402]
- (0.402,0.553]
- (0.553,0.821]

:)

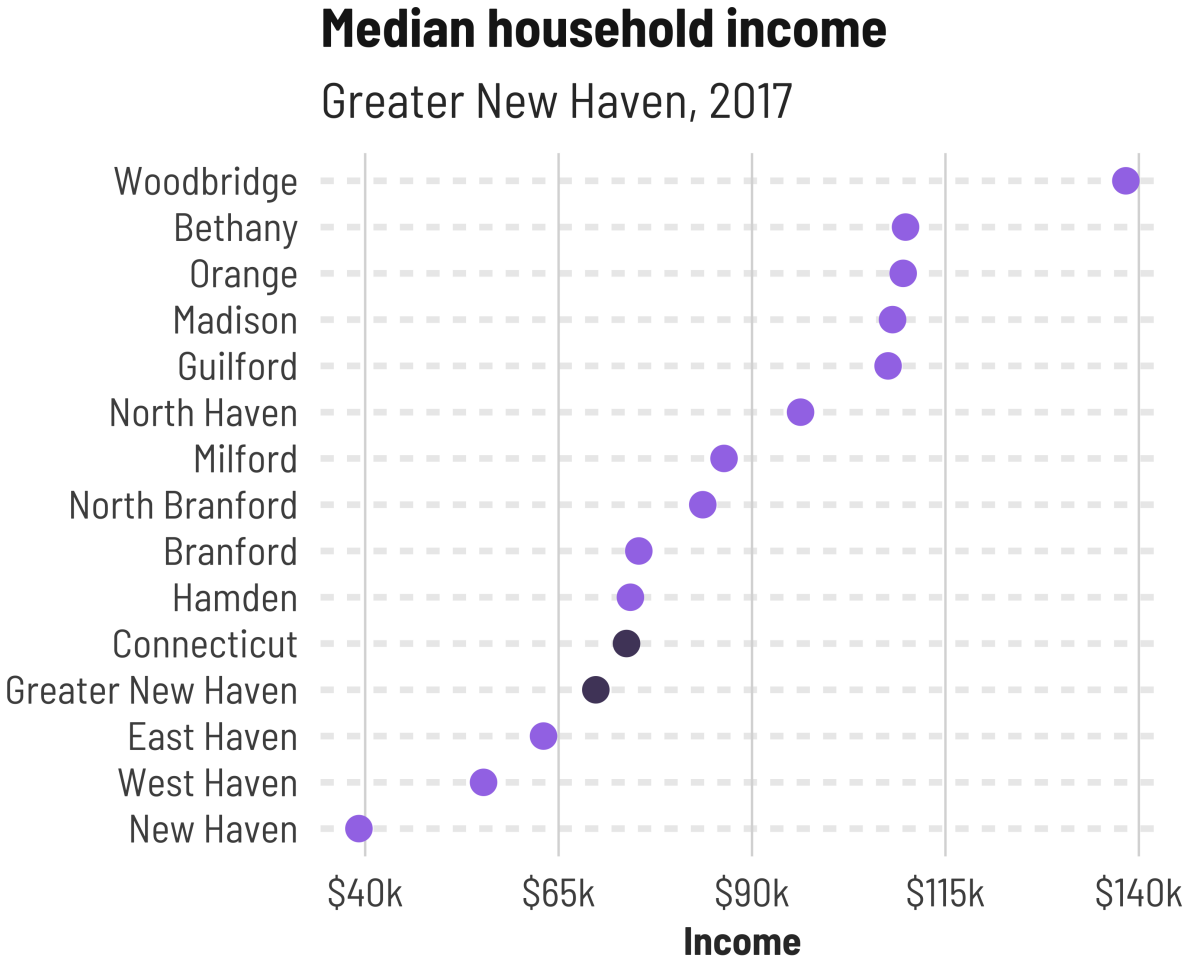
- 5 to 16%
- 16 to 28%
- 28 to 40%
- 40 to 55%
- 55 to 82%

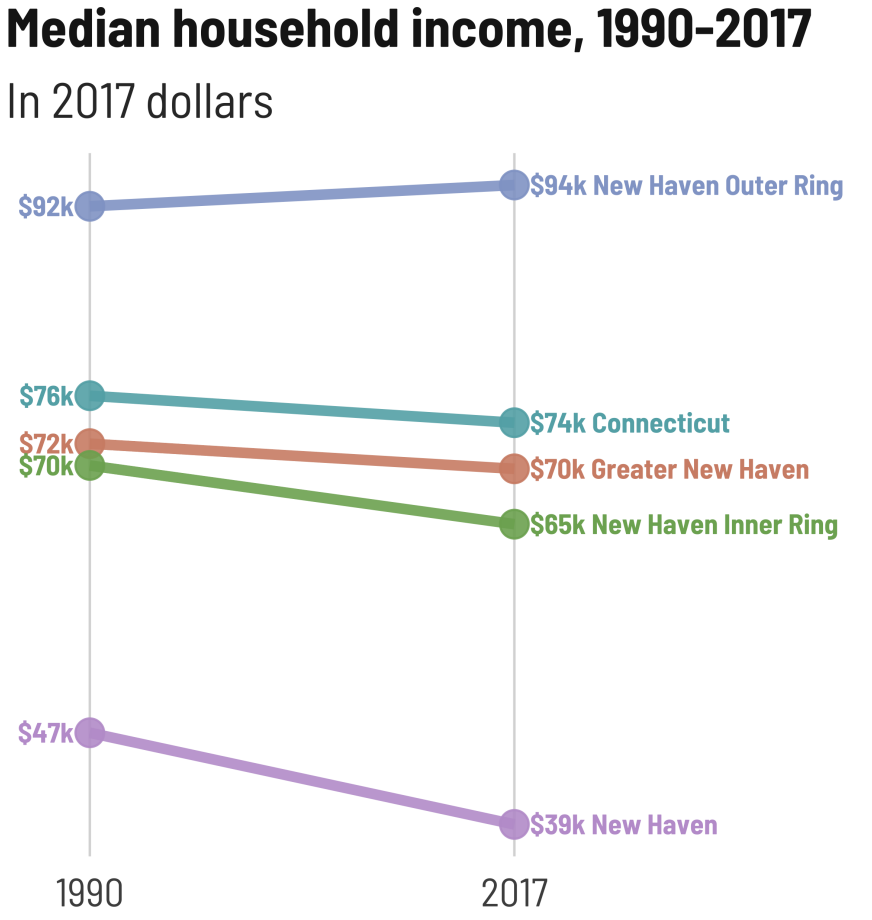
Functions: encourage good habits

```
geo_level_plot(tenure,
    value = homeownership,
    hilite = "mediumpurple1",
    title = "Homeownership rates, 2017")
```



Clean, uniform charts





Reusable datasets & references

How many times can I generate, save, and forget about the same lookup tables and shapefiles?

reference

Name	Date Modified	Size	Kind
acs_industry_codes.rds	Jun 4, 2018 at 4:10 PM	5 KB	R Data File
acs_occupation_codes.rds	Jun 4, 2018 at 4:50 PM	10 KB	R Data File
block2town.csv	Jul 5, 2018 at 4:12 PM	2.5 MB	comma...values
ct_xwalk.csv	May 26, 2018 at 3:21 PM	37.2 MB	comma...values
nhv.json	Jun 29, 2018 at 8:53 AM	42 KB	JSON file
table_2014_nh_nhood_carto_merge.geojson	Jun 28, 2018 at 6:20 PM	3.6 MB	Document
town_region_lookup.csv	Jun 7, 2018 at 1:57 PM	12 KB	comma...values
townct_37800_0000_2010_s100_census_1_shp.zip	Apr 3, 2019 at 5:18 PM	2.9 MB	ZIP archive
village_town_xwalk.csv	Mar 23, 2019 at 8:22 PM	4 KB	comma...values

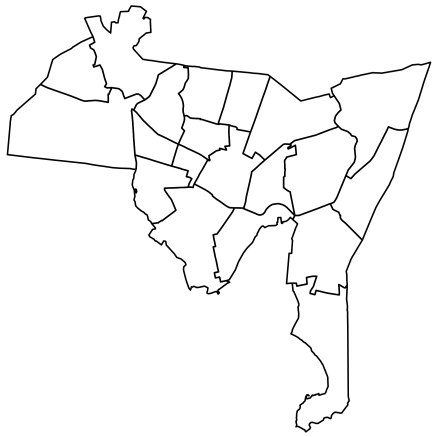
Reusable datasets & references

Much better: move those lookup tables & shapefiles to the R package

```
head(village2town, n = 5)
```

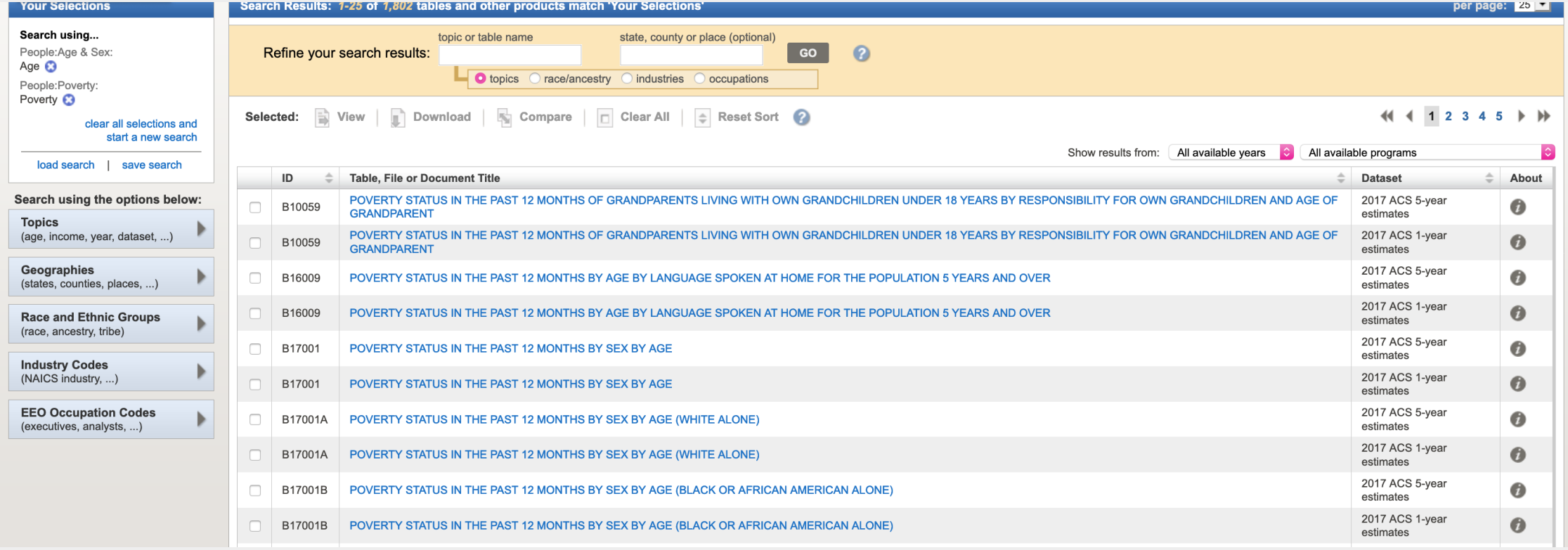
cdp_geoid	place	town_geoid	town
0902550	Baltic	0901171670	Sprague
0902690	Bantam	0900543370	Litchfield
0904945	Bethlehem Village	0900504930	Bethlehem
0906050	Blue Hills	0900305910	Bloomfield
0907345	Branford Center	0900907310	Branford

```
plot(new_haven_sf["geometry"])
```



Reusable datasets & references

Avoid the suffering of finding table numbers on FactFinder



Reusable datasets & references

Avoid the suffering of finding table numbers on FactFinder

```
basic_table_nums[["pov_age"]]
```

```
## [1] "B17024"
```

```
get_acs("county", table = basic_table_nums[["pov_age"]], state = "09")
```

Testing, debugging, documenting

What doesn't kill you makes you stronger

- Does this function do what I *think* it does?
- Are these the most important tasks for me & my coworkers?
- What might break by this time next month?
- How will this scale & remain relevant?
- What am I not thinking of yet?

Testing the `qwi_industry` function in `cwi`:

```
test_that("handles years not in API", {
  expect_warning(qwi_industry(1990:2000, industries = "23"), "earlier years are being removed")
  expect_error(qwi_industry(1990:1994, industries = "23"), "only available")
  # should only return 1996-2000
  expect_equal(nrow(suppressWarnings(qwi_industry(1991:2000, industries = "23", annual = T))), 5)
})
```

Testing, debugging, documenting

What doesn't kill you makes you stronger

- My code is amazing. Now how do I make sure someone uses it?
- If I can't explain a feature, do I really need it?
- What might someone else do wrong?
- How can I avoid "What does this do?" emails and texts?

Docs website with pkgdown

CWI 0.0.0.9000 Reference Articles

Aggregating and analyzing data

The total population data is very straightforward, as it only has one variable, `B01003_001`. The tibble returned has the GEOID, except for custom geographies like regions; the name of each geography, including the names of each region; the variable codes; estimates; margins of error at the default 90% confidence level; the geographic level, numbered in order of decreasing size; and the counties of the towns.

The race and ethnicity table will require some calculations, using the brilliantly-titled `camiller` package:

- Using `label_acs()`, join the `race` tibble with the `cwi::acs_vars` dataset to get variable labels. Oftentimes, these labels need to be separated by their `"!!"` delimeter.
- Group by the geographic level, county, and name.
- Call `camiller::add_grps()` with a list of racial groups and their labels' positions in the `label` column. This gives estimates and, optionally, margins of error for aggregates
- `camiller::calc_shares()` then gives shares of each group's estimate over the `"total"` denominator.

```
gnh_data$race %>%
  label_acs() %>%
  group_by(level, county, NAME) %>%
  add_grps(list(total = 1, white = 3, black = 4, latino = 12, other = 5:9), group
  calc_shares(group = label, denom = "total")
#> # A tibble: 90 x 6
#> # Groups:   level, county, NAME [18]
#>    level      county NAME            label  estimate  share
```

Contents

tl;dr

Package development: lots of work upfront, totally worth it

- **DataHaven:** ctdatahaven.org
- **Our side projects blog:** ct-data-haven.github.io
- **DataHaven on GitHub:** github.com/CT-Data-Haven
- **These very slides!** ct-data-haven.github.io/datadev