Nowadays (I’ve seen that word used *so* much in journal articles lately that I could not resist using it) I’m using world tile grids more frequently as the need arises to convey the state of exposure of various services at a global (country) scale. Given that necessity fosters invention it seemed that having a ggplot2 geom for world tile grids would make my life easier and also make work more efficient.

To that end, there’s a nascent ggplot2 extension package for making world tile grids called (uncreatively) worldtilegrid.

But we’ll walk through another one here and work with life expectancy data curated by [Our World in Data](https://ourworldindata.org/life-expectancy). Rather than draw out this post to a less tenable length with an explanation of how to pull the XHR JSON data into R, just grab the CSV linked with the visualization on that page and substitute the path in the code for where you stored it.

We do need to clean up this data a bit since it has some issues. Let’s do that and carve out some slices into two new data frames so we can work with the most recent curated year and some historical data:

library(hrbrthemes)

library(worldtilegrid)

library(tidyverse)

cols(

Entity = col\_character(),

Code = col\_character(),

Year = col\_integer(),

`Life expectancy (Clio-Infra up to 1949; UN Population Division for 1950 to 2015)` = col\_double()

) -> le\_cols

read\_csv("~/Downloads/life-expectancy.csv", col\_types = le\_cols) %>%

set\_names(c("country", "iso3c", "year", "life\_expectancy")) -> lifexp

# clean it up a bit since it's not great data and bust it into groups

# that match the vis on the site vs use continuous raw data, esp since

# the data is really just estimates

filter(lifexp, !is.na(iso3c)) %>%

filter(nchar(iso3c) == 3) %>%

mutate(

grp = cut(

x = life\_expectancy,

breaks = c(10, 20, 30, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85),

include.lowest = TRUE

)

) -> lifexp

last\_year <- filter(lifexp, year == last(year))

a\_few\_years <- filter(lifexp, year %in% c(seq(1900, 2015, 20), 2015))

Now we can use it to make some cartograms.

Making World Tile Grid Grids

The Premise

For this particular use-case, I sifted through our internet scan data and classified a series of device families from their telnet banners then paired that with our country-level attribution data for each IPv4 address. I’m not generally “a fan” of rolling things up at a country level, but since many (most) of these devices are residential or small/medium-business routers, country-level attribution has some merit.

But, I’m also not a fan of country-level choropleths when it comes to “cyber” nor am I wont to area-skewed cartograms since most folks still cannot interpret them. Both of those take up a ton of screen real estate, too, espeically if you have more than one of them. Yet, I wanted to show a map-like structure without resorting to Hilbert IPv4 heatmaps since they are neither very readable by a general audience and become skewed when you have to move up from a 1 pixel == 1 Class C network block.

I think the tile grid is a great compromise since it avoids the “area”and projection skewness confusion that regular global choropleths cause while still preserving geographic & positional proximity. Sure, they’ll take some getting used to by casual readers, but I felt it was the best of all the tradeoffs.

The Setup

Here’s the data:

library(here)

library(hrbrthemes)

library(tidyverse)

wtg <- read\_csv("https://gist.githubusercontent.com/maartenzam/787498bbc07ae06b637447dbd430ea0a/raw/9a9dafafb44d8990f85243a9c7ca349acd3a0d07/worldtilegrid.csv")

glimpse(wtg)

## Observations: 192

## Variables: 11

## $ name "Afghanistan", "Albania", "Algeria", "Angola",...

## $ alpha.2 "AF", "AL", "DZ", "AO", "AQ", "AG", "AR", "AM"...

## $ alpha.3 "AFG", "ALB", "DZA", "AGO", "ATA", "ATG", "ARG...

## $ country.code "004", "008", "012", "024", "010", "028", "032...

## $ iso\_3166.2 "ISO 3166-2:AF", "ISO 3166-2:AL", "ISO 3166-2:...

## $ region "Asia", "Europe", "Africa", "Africa", "Antarct...

## $ sub.region "Southern Asia", "Southern Europe", "Northern ...

## $ region.code "142", "150", "002", "002", NA, "019", "019", ...

## $ sub.region.code "034", "039", "015", "017", NA, "029", "005", ...

## $ x 22, 15, 13, 13, 15, 7, 6, 20, 24, 15, 21, 4, 2...

## $ y 8, 9, 11, 17, 23, 4, 14, 6, 19, 6, 7, 2, 9, 8,...

routers <- read\_csv(here::here("data", "routers.csv"))

routers

## # A tibble: 453,027 x 3

## type country\_name country\_code

##

## 1 mikrotik Slovak Republic SK

## 2 mikrotik Czechia CZ

## 3 mikrotik Colombia CO

## 4 mikrotik Bosnia and Herzegovina BA

## 5 mikrotik Czechia CZ

## 6 mikrotik Brazil BR

## 7 mikrotik Vietnam VN

## 8 mikrotik Brazil BR

## 9 mikrotik India IN

## 10 mikrotik Brazil BR

## # ... with 453,017 more rows

distinct(routers, type) %>%

arrange(type) %>%

print(n=11)

## # A tibble: 11 x 1

## type

##

## 1 asus

## 2 dlink

## 3 huawei

## 4 linksys

## 5 mikrotik

## 6 netgear

## 7 qnap

## 8 tplink

## 9 ubiquiti

## 10 upvel

## 11 zte

So, we have 11 different device families under assault by “VPNFilter” and I wanted to show the global distribution of them. Knowing the compact world tile grid would facet well, I set off to make it happen.

Let’s get some decent names for facet labels:

real\_names <- read\_csv(here::here("data", "real\_names.csv"))

real\_names

## # A tibble: 11 x 2

## type lab

##

## 1 asus Asus Device

## 2 dlink D-Link Devices

## 3 huawei Huawei Devices

## 4 linksys Linksys Devices

## 5 mikrotik Mikrotik Devices

## 6 netgear Netgear Devices

## 7 qnap QNAP Devices

## 8 tplink TP-Link Devices

## 9 ubiquiti Ubiquiti Devices

## 10 upvel Upvel Devices

## 11 zte ZTE Devices

Next, we need to summarise our scan results and pair it up the world tile grid data and our real names:

count(routers, country\_code, type) %>% # summarise the data into # of device familes per country

left\_join(wtg, by = c("country\_code" = "alpha.2")) %>% # join them up on the common field

filter(!is.na(alpha.3)) %>% # we only want countries on the grid and maxmind attributes some things to meta-regions and anonymous proxies

left\_join(real\_names) -> wtg\_routers

glimpse(wtg\_routers)

## Observations: 629

## Variables: 14

## $ country\_code "AE", "AE", "AE", "AF", "AF", "AF", "AG", "AL"...

## $ type "asus", "huawei", "mikrotik", "huawei", "mikro...

## $ n 1, 12, 70, 12, 264, 27, 1, 941, 2081, 7, 2, 1,...

## $ name "United Arab Emirates", "United Arab Emirates"...

## $ alpha.3 "ARE", "ARE", "ARE", "AFG", "AFG", "AFG", "ATG...

## $ country.code "784", "784", "784", "004", "004", "004", "028...

## $ iso\_3166.2 "ISO 3166-2:AE", "ISO 3166-2:AE", "ISO 3166-2:...

## $ region "Asia", "Asia", "Asia", "Asia", "Asia", "Asia"...

## $ sub.region "Western Asia", "Western Asia", "Western Asia"...

## $ region.code "142", "142", "142", "142", "142", "142", "019...

## $ sub.region.code "145", "145", "145", "034", "034", "034", "029...

## $ x 20, 20, 20, 22, 22, 22, 7, 15, 15, 15, 20, 20,...

## $ y 10, 10, 10, 8, 8, 8, 4, 9, 9, 9, 6, 6, 6, 6, 1...

## $ lab "Asus Device", "Huawei Devices", "Mikrotik Dev...

Then, plot it:

ggplot(wtg\_routers, aes(x, y, fill=n, group=lab)) +

geom\_tile(color="#b2b2b2", size=0.125) +

scale\_y\_reverse() +

viridis::scale\_fill\_viridis(name="# Devices", trans="log10", na.value="white", label=scales::comma) +

facet\_wrap(~lab, ncol=3) +

coord\_equal() +

labs(

x=NULL, y=NULL,

title = "World Tile Grid Per-country Concentration of\nSeriously Poorly Configured Network Devices",

subtitle = "Device discovery based on in-scope 'VPNFilter' vendor device banner strings",

caption = "Source: Rapid7 Project Sonar & Censys"

) +

theme\_ipsum\_rc(grid="") +

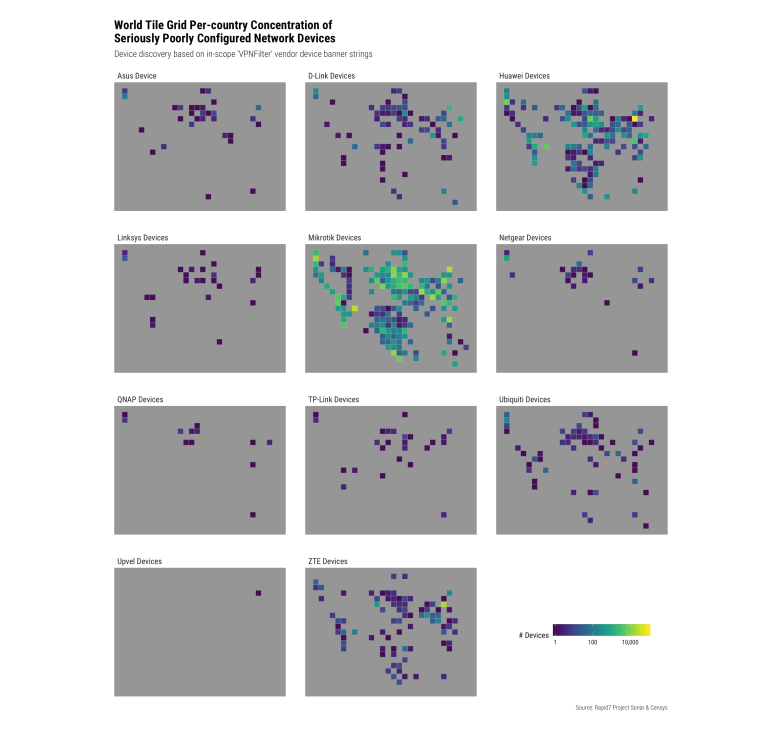
theme(panel.background = element\_rect(fill="#969696", color="#969696")) +

theme(axis.text=element\_blank()) +

theme(legend.direction="horizontal") +

theme(legend.key.width = unit(2, "lines")) +

theme(legend.position=c(0.85, 0.1))



Doh! We forgot to ensure we had data for every country. Let’s try that again:

count(routers, country\_code, type) %>%

complete(country\_code, type) %>%

filter(!is.na(country\_code)) %>%

left\_join(wtg, c("country\_code" = "alpha.2")) %>%

filter(!is.na(alpha.3)) %>%

left\_join(real\_names) %>%

complete(country\_code, type, x=unique(wtg$x), y=unique(wtg$y)) %>%

filter(!is.na(lab)) %>%

ggplot(aes(x, y, fill=n, group=lab)) +

geom\_tile(color="#b2b2b2", size=0.125) +

scale\_y\_reverse() +

viridis::scale\_fill\_viridis(name="# Devices", trans="log10", na.value="white", label=scales::comma) +

facet\_wrap(~lab, ncol=3) +

coord\_equal() +

labs(

x=NULL, y=NULL,

title = "World Tile Grid Per-country Concentration of\nSeriously Poorly Configured Network Devices",

subtitle = "Device discovery based on in-scope 'VPNFilter' vendor device banner strings",

caption = "Source: Rapid7 Project Sonar & Censys"

) +

theme\_ipsum\_rc(grid="") +

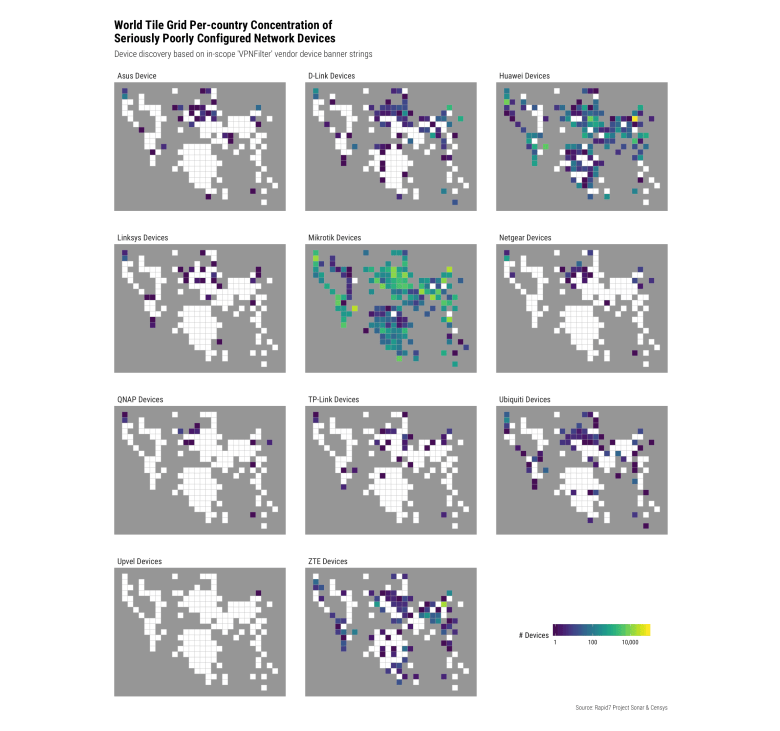
theme(panel.background = element\_rect(fill="#969696", color="#969696")) +

theme(axis.text=element\_blank()) +

theme(legend.direction="horizontal") +

theme(legend.key.width = unit(2, "lines")) +

theme(legend.position=c(0.85, 0.1))



That’s better.

We take advantage of ggplot2’s ability to facet and just ensure we have complete (even if NA) tiles for each panel.

**Using the World Tile Grid geom**

The idea was to simplify the API down to the caller just needing to specify some country names/ISO[23]Cs (to a country aesthetic) and a value (to a fill aesthetic) and let the package do the rest. To that end, here's how to display the life expectancy data for 2015:

ggplot(last\_year) +

geom\_wtg(aes(country = iso3c, fill = grp), border\_col = "#2b2b2b") + # yep, you can sum up the entire blog post to this one line

viridis::scale\_fill\_viridis(

name = "Life Expectancy", discrete = TRUE, na.value=alpha(ft\_cols$gray, 1/10),

drop = FALSE

) +

coord\_equal() +

labs(

x=NULL, y=NULL,

title = "Life expectancy, 2015",

subtitle = "Shown is period life expectancy at birth. This corresponds to an estimate of the average number\nof years a newborn infant would live if prevailing patterns of mortality at the time of its birth were to\nstay the same throughout its life",

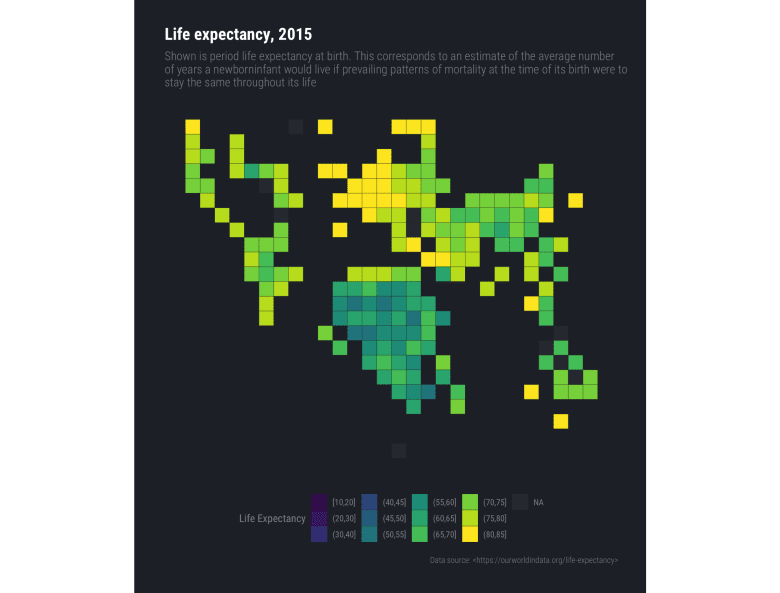
caption = "Data source: "

) +

theme\_ft\_rc(grid="") +

theme(axis.text=element\_blank()) +

theme(legend.position = "bottom")

[](https://rud.is/b/2018/08/27/simplifying-world-tile-grid-creation-with-geom_wtg/le-wtg-01/)

You can add labels via geom\_text() and use the wtg stat for it as it provides a number of computed variables you can work with:

* x,y: the X,Y position of the tile
* name: Country name (e.g. Afghanistan)
* country.code: ISO2C country code abbreviation (e.g. AF)
* iso\_3166.2: Full ISO 3166 2-letter abbreviation code (e.g. ISO 3166-2:AF)
* region: Region name (e.g. Asia)
* sub.region: Sub-region name (e.g. Southern Asia)
* region.code: Region code (e.g. 142)
* sub.region.code: Sub-region code (e.g. 034)

Labeling should be a deliberate decision and used sparingly/with care.

**Easier World Tile Grid Facets**

Making faceted charts is also straightforward. Just use ggplot2's faceting subsytem:

ggplot(a\_few\_years) +

geom\_wtg(aes(country = iso3c, fill = grp), border\_col = "#2b2b2b") +

viridis::scale\_fill\_viridis(

name = "Life Expectancy", discrete = TRUE, na.value=alpha(ft\_cols$gray, 1/10),

drop = FALSE

) +

coord\_equal() +

labs(

x=NULL, y=NULL,

title = "Life expectancy, 1900-2015 (selected years)",

subtitle = "Shown is period life expectancy at birth. This corresponds to an estimate of the average number of years a newborn infant\nwould live if prevailing patterns of mortality at the time of its birth were to stay the same throughout its life",

caption = "Data source: "

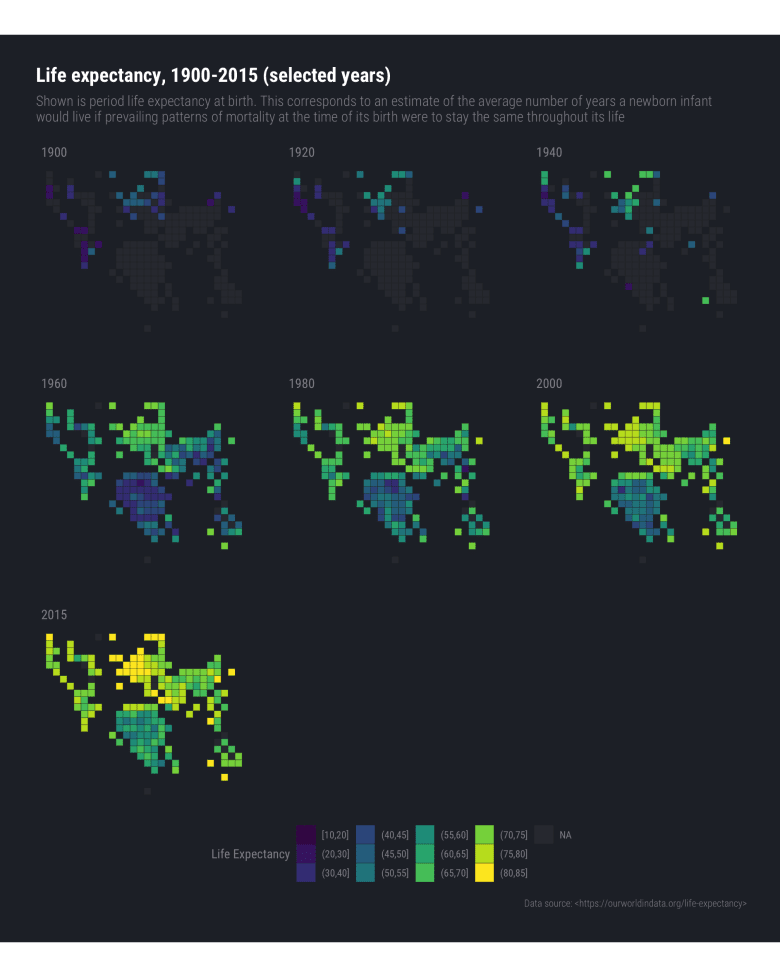
) +

facet\_wrap(~year) +

theme\_ft\_rc(grid="") +

theme(axis.text=element\_blank()) +

theme(legend.position = "bottom")

[](https://rud.is/b/2018/08/27/simplifying-world-tile-grid-creation-with-geom_wtg/le-wtg-02/)

(You may want to open that one up in a separate tab/window.)

Animation might have been a better choice than facets (which is an exercise left to the reader ).

**FIN**

The package API is still in "rapidly changing" mode, but feel free to kick the tyres and file issues or PRs on your community coding platform of choice as needed.