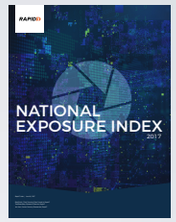
Using R for Disease Surveillance, Enabling Citizen Data Science & Cybersecurity+safety Tips for Scientists

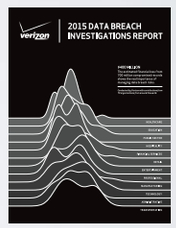
ISDS R Group for Biosurveillance
2018-01-24

I'd like to thank everyone for attending the webinar today. I went through the recordings of previous webinars and you've had some great presenters on some really interesting topics (I especially liked Shirin Glander's talk on machine learning models).

ABOUT://ME

- 20+ years in cybersecurity
- Author of 70+ R packages (a dozen or so on CRAN)
- Contributor to rOpenSci
- Co-author of Data-Driven Security (Wiley Press)
- Chief Data Scientist, Rapid7
- Former chief researcher & team lead, Verizon Data Breach Investigations Report





You got most of this in the bio. The most important thing on here is the Captain America shield. If you ever need to find me, just look for the shield.

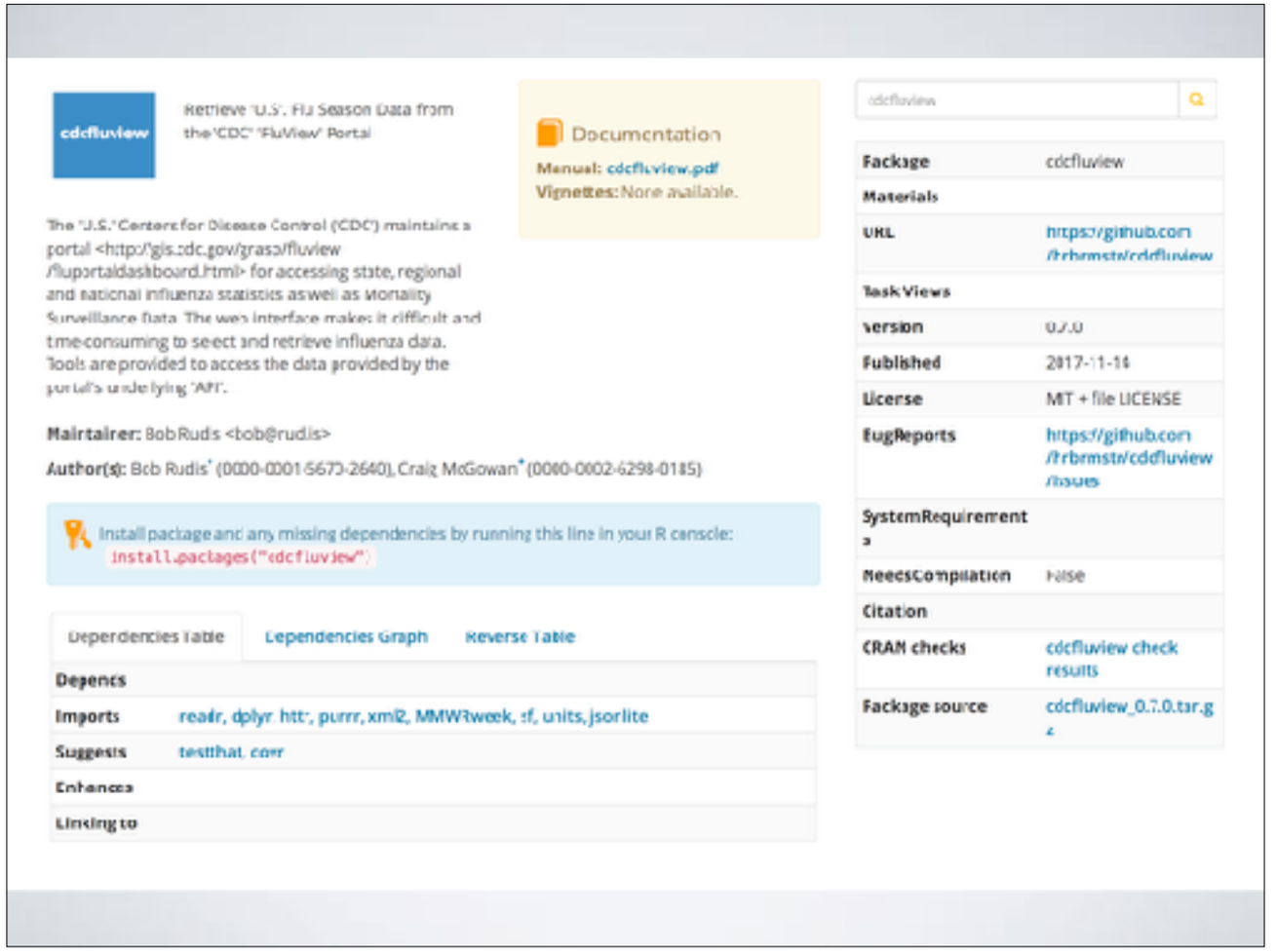
DISCLAIMER: I'm not an epidemiologist.

I don't even play one on TV like this gentleman.

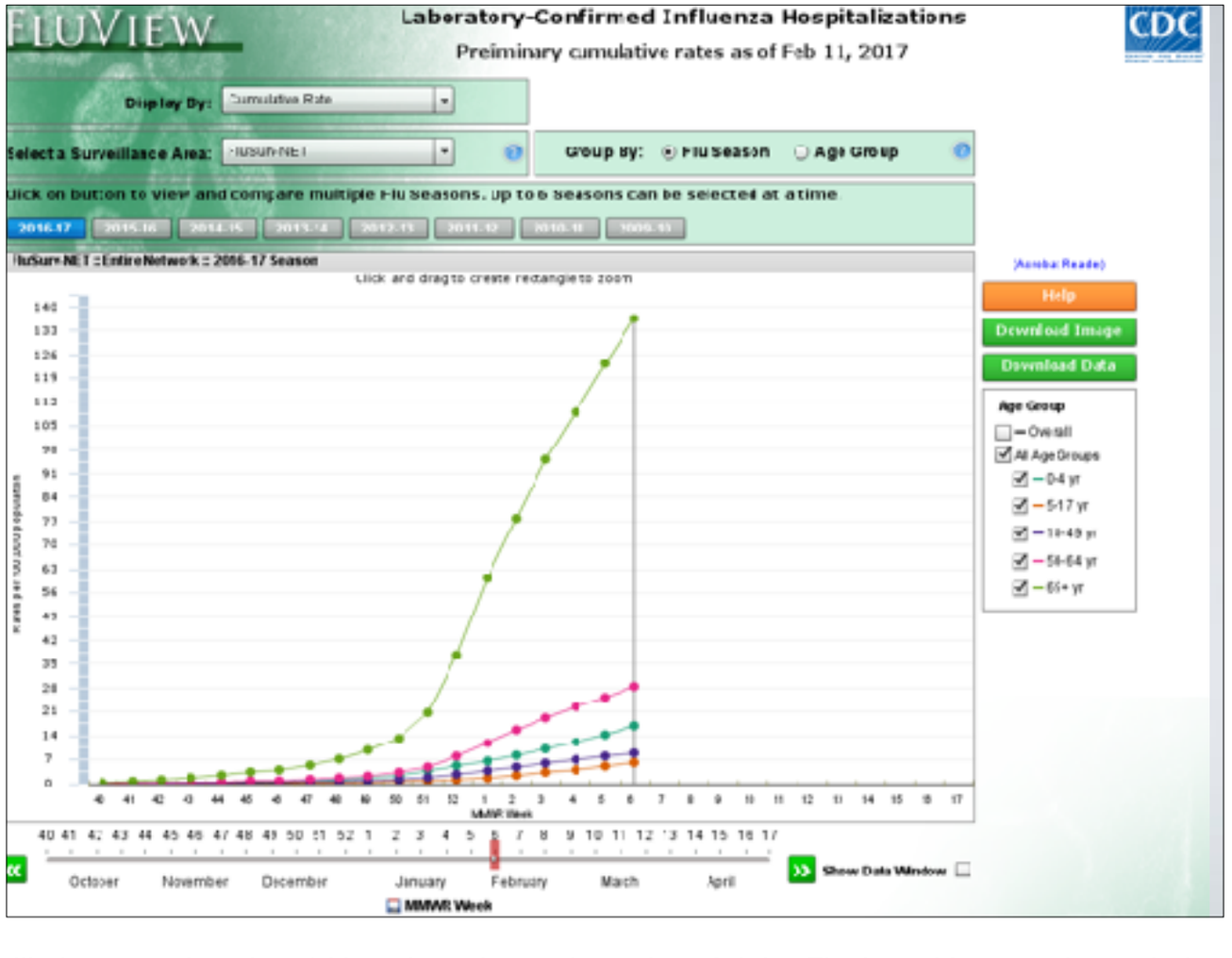
So, by now you may be wondering what prompted a cybersecurity professional to create a package that enables reproducible research with and dashboard creation of CDC influenza surveillance data. Hopefully I can help answer that question and also introduce you to some of the core features of the package.

😷 CDCFLUVIEW 📦

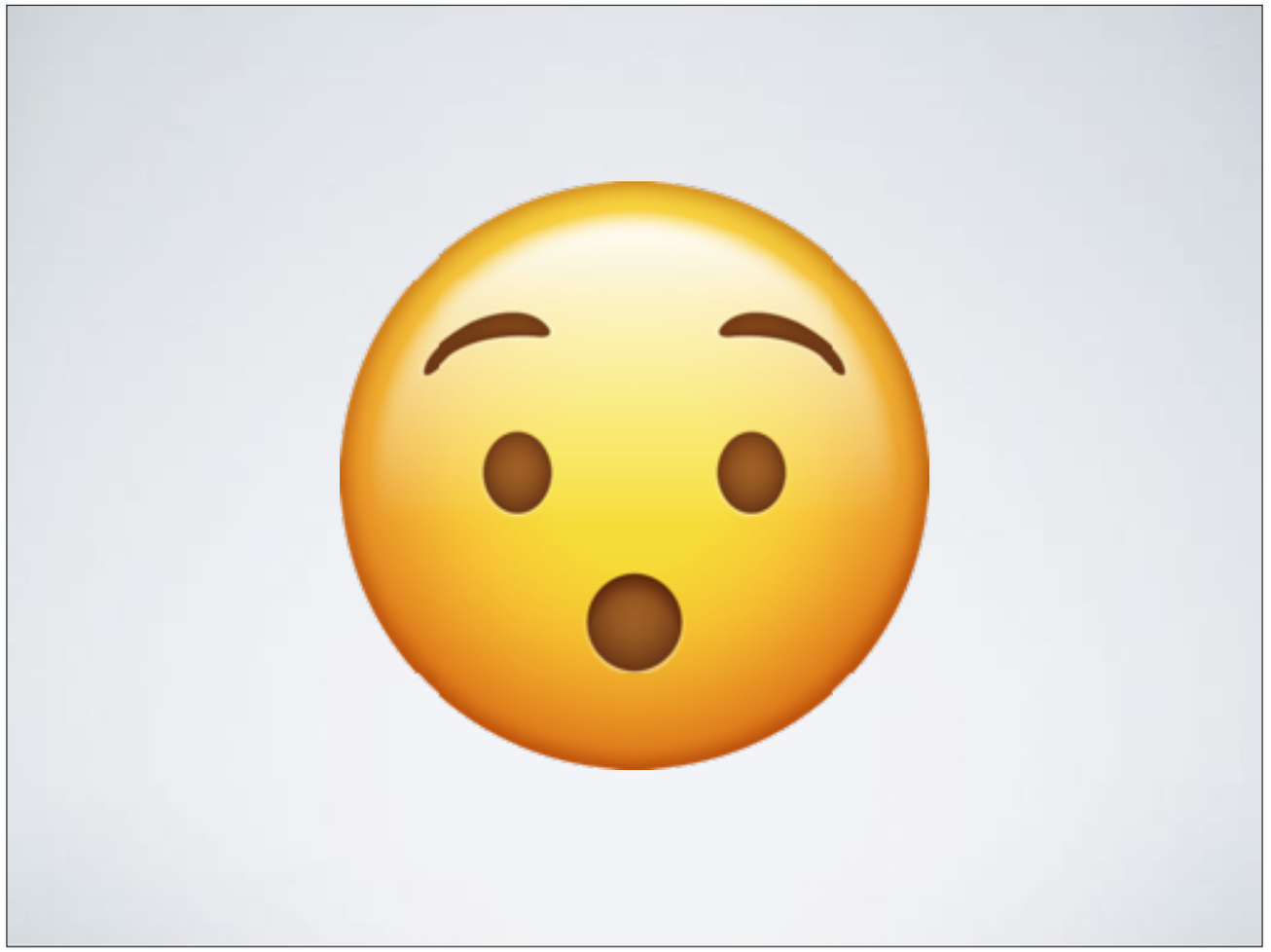
What is cdcfluview?



It's very likely that all of you are intimately familiar with the CDC's FluView portal. They provide a wealth of information — and some questionable visualizations — on influenza and other influenza like illnesses. The main focus of the cdcfluview R package is to provide a streamlined query interface to all the data behind the FluView portal but there are also tools to work with some of the geo-data and it has sibling packages that make it straightforward to quickly produce visualizations from the data.



If you are familiar with CDC FluView, you likely remember the old interface. It was based on Adobe Flash and I was trying to help a non-cyber colleague script getting to the underlying data. Rather than just make a one-off script, I decided to dive into the hidden API and turn it into a full-featured R package. I'll talk about that more in a bit, but I have a bit of a package addiction when it comes to R. Packages are the easiest way to bundle up operations and documentation and a good package enables the user to forget about the mechanics of data access and just focus on the analysis. So, while I'm not an epidemiologist, I am fairly adept at dissecting hidden APIs (hacking experience comes in handy in many ways) and making them useful. I thought I was making one person's life easier.

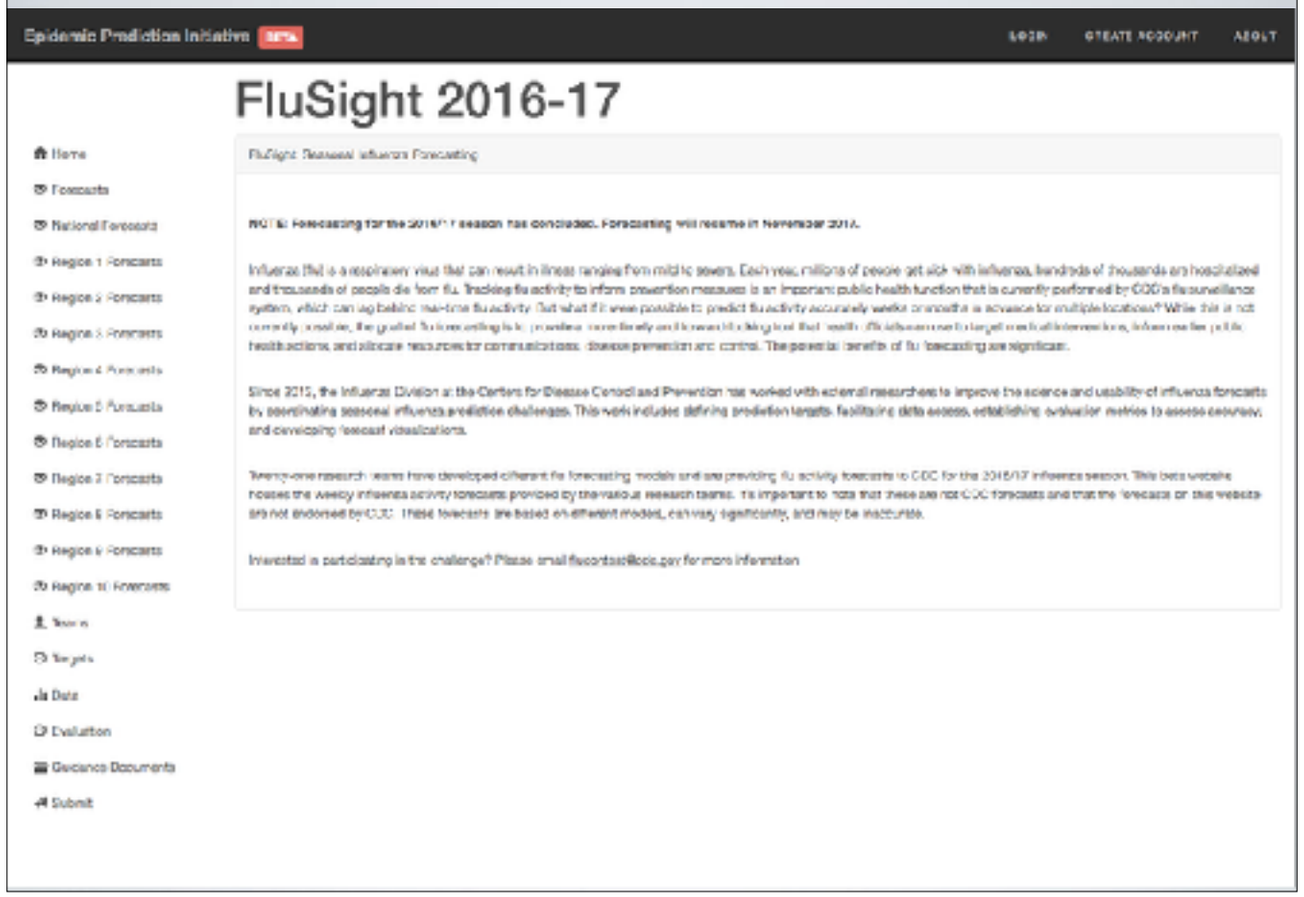


But, it seems others figured out that I made the package (which surprised me since it's fairly niche package)

https://umnflu.wordpress.com/2017/11/06/retrieving-ili-from-cdcfluview/

The folks at UMN use it for their real-time dashboard (totally no pressure to ensure the package works. nope, not at all)

https://predict.phiresearchlab.org/



I also discovered it's suggested by (but not endorsed or affiliated with) the Epidemic Prediction Initiative (if you click on Data you'll see instructions)

https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=6780&context=etd

IOWA STATE UNIVERSITY
Digital Repository

Graduate Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

2016

Bayesian models and inferential methods for forecasting disease outbreak severity

Nicholas Lorens Michaud
Iowa State University

There are a few citations of it (but papers don't really cite R packages as much as they should) so I have no real idea the extent of the usage.

At this point, you're likely thinking "Just show me the package". So let's go!

Retrieve ILINet Surveillance Data

Description

The CDC FluView Portal provides in-season and past seasons' national, regional, and state-level outpatient illness and viral surveillance data from both ILINet (Influenza-like Illness Surveillance Network) and WHO/NREVSS (National Respiratory and Enteric Virus Surveillance System).

Usage

```
ilinet(region = c("national", "hhs", "census","state"), years = NULL)
```

Arguments

`region`	one of `"national"`, `"hhs"`, `"census"`, or `"state"`
`years`	a vector of years to retrieve data for (i.e. `2014` for CDC flu season 2014-2015). CDC has data for this API going back to 1997. Default value (`NULL`) means retrieve **all** years. NOTE: if you happen to specify a 2-digit season value (i.e. `57` == 2017-2018) the function is smart enough to retrieve by season ID vs convert that to a year.

```r
# stable / CRAN
install.packages(cdcfluview)

# bleeding edge
devtools::install_github("hrbrmstr/cdcfluview")
```

```
install.packages(ggalt)
install.packages(statebins)
install.packages(hrbrthemes)
```

these are other packages of mine that I'll be using in some of the examples (I'm not going to overwhelm you with code). They're all on CRAN but each has some cutting edge functionality on GitHub (I also try to not overwhelm CRAN folks with package updates since they're volunteers). There's a link to a GitHub repository for this talk that has all of the packages being used in it, so there's no need to frantically copy everything from the screen.

```
library(ggalt)
library(cdcfluview)
library(hrbrthemes)
library(tidyverse)

ili <- ilinet("national", years = 2013:2018)
```

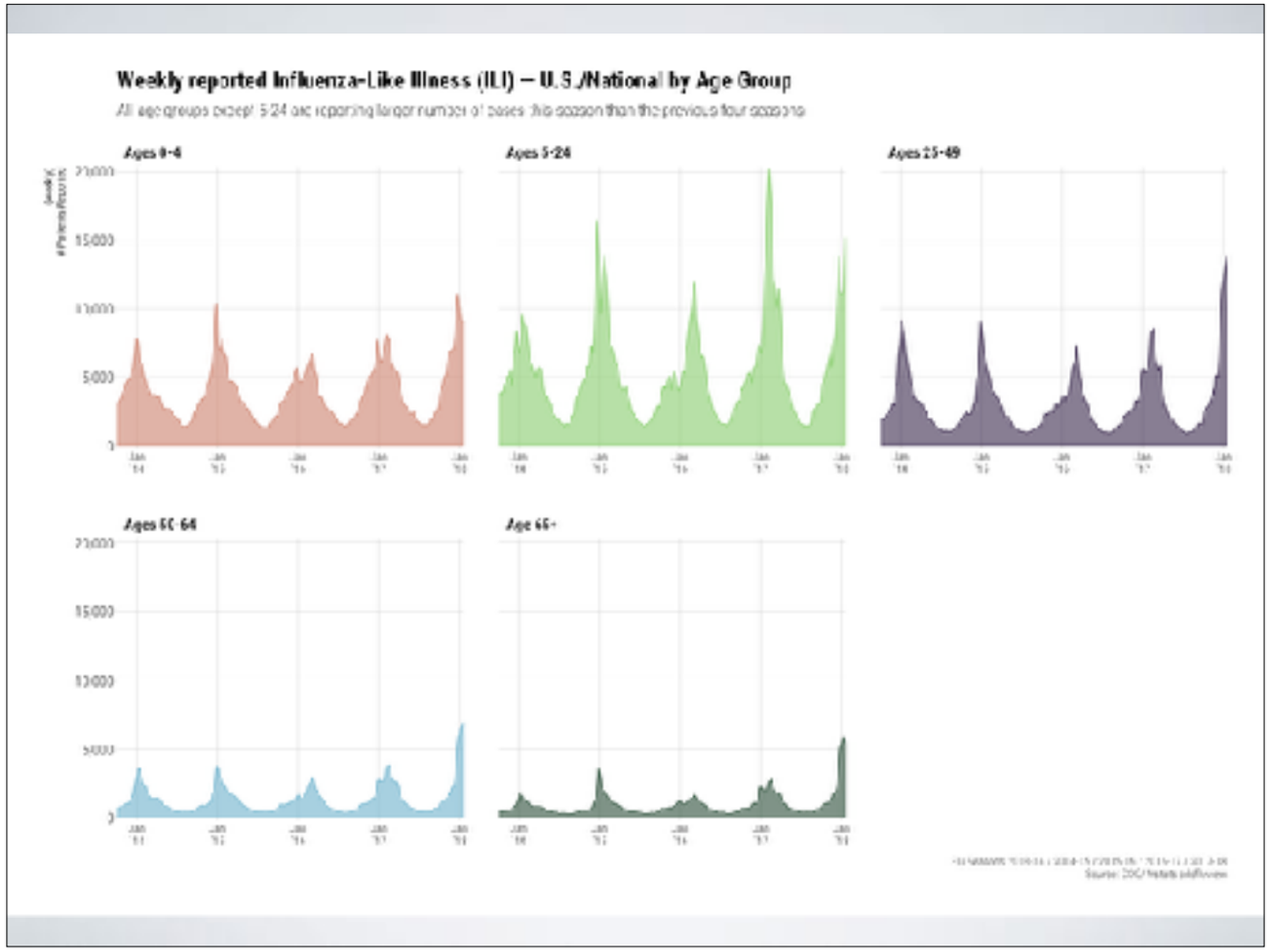
```
## Observations: 224
## Variables: 16
## $ region_type       <chr> "National", "National", "National", "National...
## $ region            <chr> "National", "National", "National", "National...
## $ year              <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 201...
## $ week              <int> 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 5...
## $ weighted_ili      <dbl> 1.15746, 1.27184, 1.31623, 1.37444, 1.46781, ...
## $ unweighted_ili    <dbl> 1.13249, 1.21121, 1.25682, 1.37097, 1.42956, ...
## $ age_0_4           <dbl> 2974, 3276, 3483, 3930, 4045, 4434, 4643, 483...
## $ age_25_49         <dbl> 1840, 1924, 2010, 2292, 2354, 2511, 2716, 308...
## $ age_25_64         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ age_5_24          <dbl> 3769, 3925, 3880, 4484, 4642, 5062, 5000, 557...
## $ age_50_64         <dbl> 638, 762, 785, 816, 930, 953, 990, 1083, 1059...
## $ age_65            <dbl> 456, 533, 500, 514, 531, 538, 578, 583, 639, ...
## $ ilitotal          <dbl> 9677, 10420, 10658, 12036, 12502, 13498, 1392...
## $ num_of_providers  <dbl> 1960, 1990, 2018, 2027, 2028, 2017, 2054, 201...
## $ total_patients    <dbl> 854487, 860298, 848016, 877917, 874537, 87884...
## $ week_start        <date> 2013-10-07, 2013-10-14, 2013-10-21, 2013-10-...
```

```
hhs <- ilinet("hhs", years = 2013:2018)
```

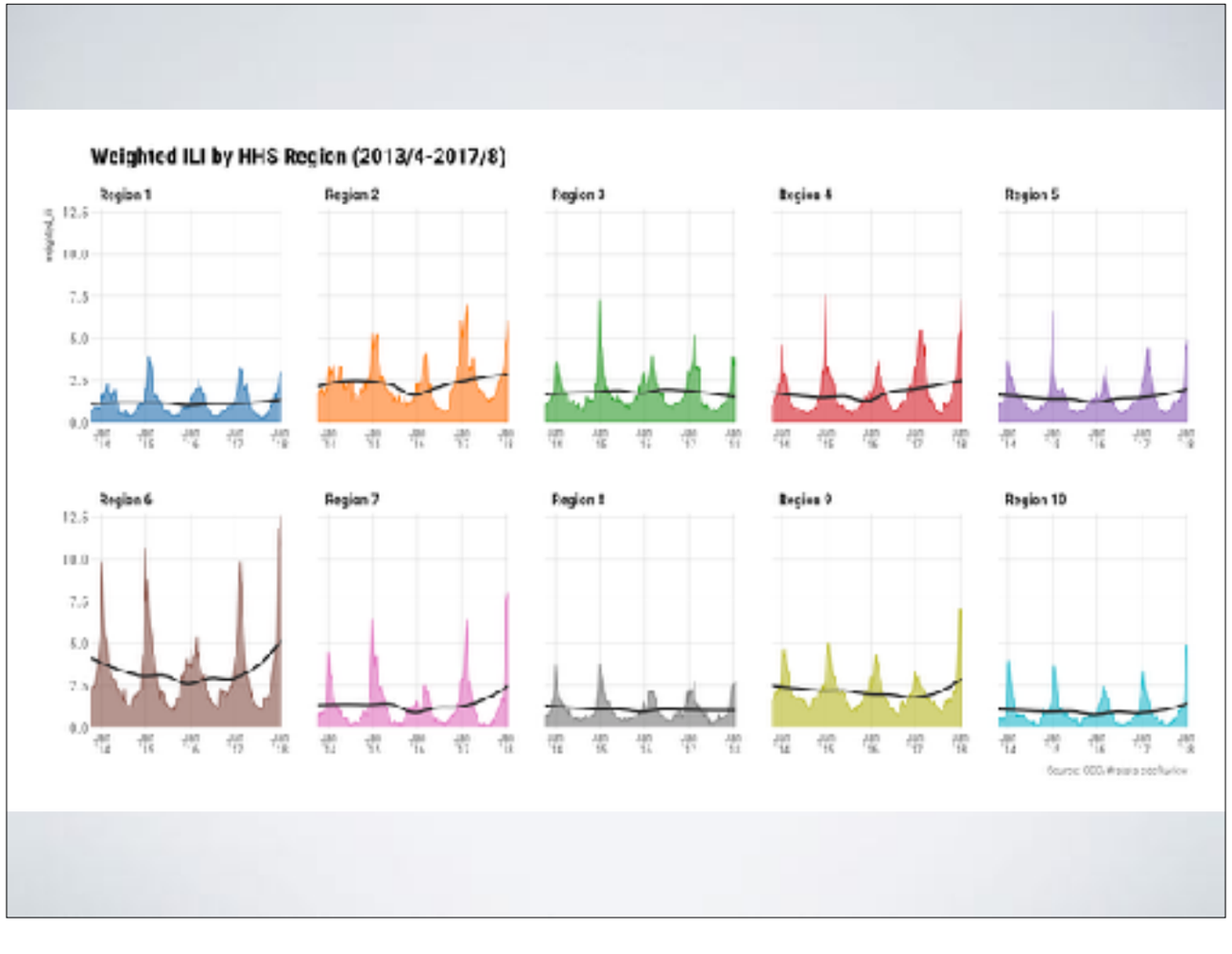
```
Observations: 2,240
Variables: 16
$ region_type       <chr> "HHS Regions", "HHS Regions", "HHS Regions", "HH...
$ region            <fctr> Region 1, Region 2, Region 3, Region 4, Region ...
$ year              <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ...
$ week              <int> 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 41, 41, ...
$ weighted_ili      <dbl> 0.621395, 1.607510, 0.926323, 0.943459, 0.885838...
$ unweighted_ili    <dbl> 0.615189, 1.431470, 1.094000, 1.020040, 1.139130...
$ age_0_4           <dbl> 117, 632, 401, 432, 402, 585, 48, 111, 234, 12, ...
$ age_25_49         <dbl> 44, 321, 293, 249, 207, 418, 39, 74, 178, 17, 39...
$ age_25_64         <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
$ age_5_24          <dbl> 168, 644, 465, 666, 435, 709, 71, 178, 399, 34, ...
$ age_50_64         <dbl> 19, 120, 75, 62, 110, 120, 17, 28, 79, 8, 15, 12...
$ age_65            <dbl> 20, 75, 55, 44, 76, 78, 7, 17, 82, 2, 16, 68, 46...
$ ilitotal          <dbl> 368, 1792, 1289, 1453, 1230, 1910, 182, 408, 972...
$ num_of_providers  <dbl> 176, 222, 233, 322, 270, 235, 90, 119, 246, 47, ...
$ total_patients    <dbl> 59819, 125186, 117824, 142445, 107977, 91046, 53...
$ week_start        <date> 2013-10-07, 2013-10-07, 2013-10-07, 2013-10-07,...
```

```
update_geom_font_defaults(font_rc)
theme_set(theme_ipsum_rc(grid = "XY", strip_text_face = "bold"))

select(ili, week_start, starts_with("age")) %>%
  select(-age_25_64) %>%
  gather(group, ct, -week_start) %>%
  mutate(group = factor(group, levels = c("age_0_4", "age_5_24", "age_25_49", "age_50_64", "age_65"),
                        labels = c("Ages 0-4", "Ages 5-24", "Ages 25-49", "Ages 50-64", "Age 65+"))) %>%
  ggplot(aes(week_start, ct, group = group)) +
  stat_xspline(geom = "area", aes(color = group, fill = group), size = 2/5, alpha = 2/3) +
  scale_x_date(expand = c(0,0), date_labels = "%b\n`%y") +
  scale_y_comma() +
  scale_color_ipsum() +
  scale_fill_ipsum() +
  labs(
    x = NULL, y = "(weekly)\n# Patients Reported",
    title="Weekly reported Influenza-Like Illness (ILI) - U.S./National by Age Group",
    subtitle="All age groups except 5-24 are reporting larger number of cases this season than the previous four seasons",
    caption="Flu Seasons 2013-14 / 2014-15 / 2015-16 / 2016-17 / 2017-18\nSource: CDC/#rstats cdcfluview"
  ) +
  facet_wrap(~group, scales = "free_x", nrow = 2) +
  theme(axis.text.x = element_text(size = 9)) +
  theme(legend.position = "none")
```


Weekly reported Influenza-Like Illness (ILI) — U.S./National by Age Group
All age groups except 5-24 are reporting larger numbers of cases this season than the previous four seasons
Ages 0-4
Ages 5-24
Ages 25-49
Ages 50-64
Age 65+
15,000
5,000
0

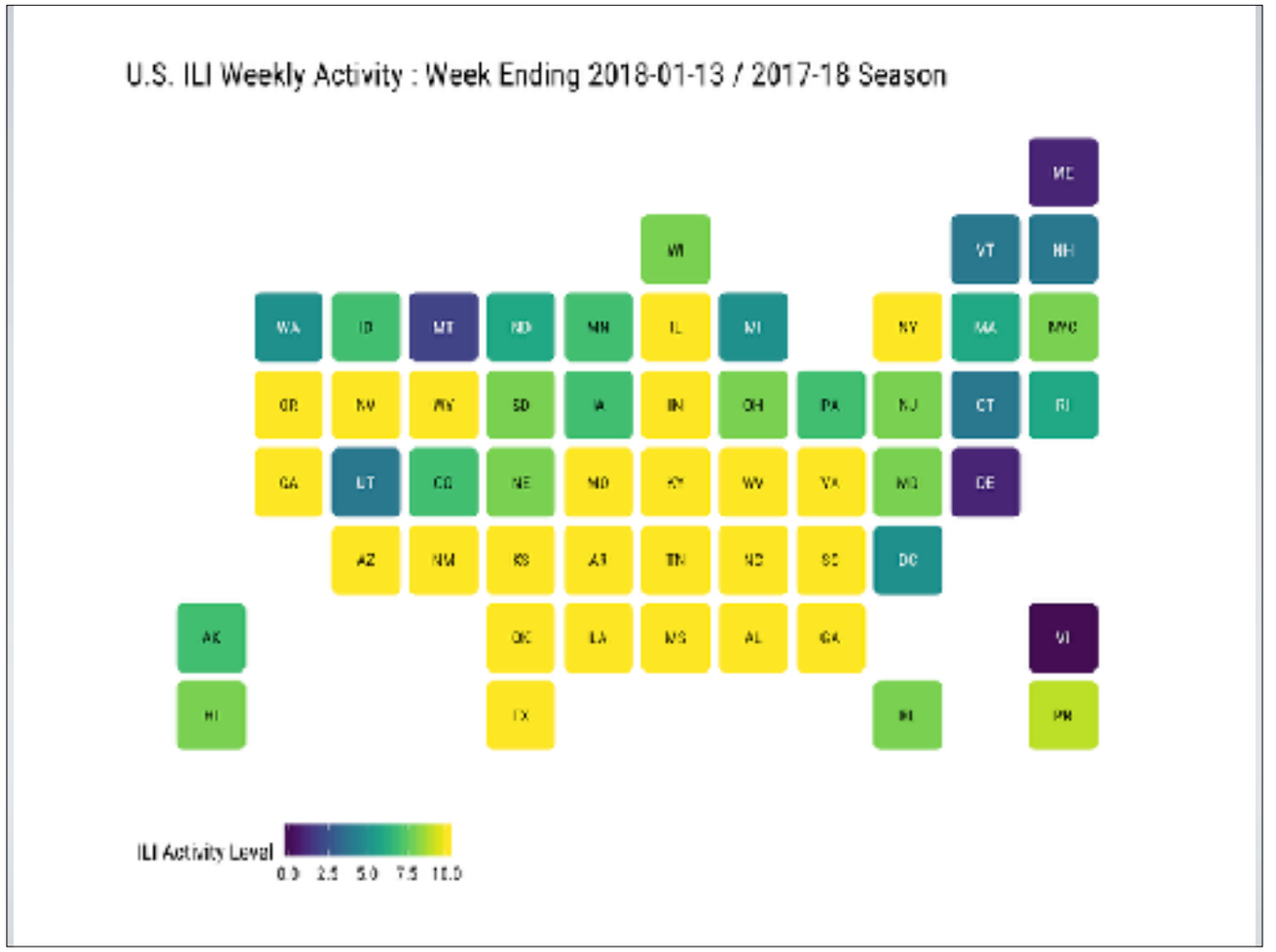
```
ggplot(hhs, aes(week_start, weighted_ili, group=region)) +
  stat_xspline(geom="area", aes(color=region, fill=region),
               size=2/5, alpha=2/3) +
  geom_smooth(se=FALSE, size=1, color="#2b2b2b") +
  scale_x_date(expand=c(0,0), date_labels="%b\n`%y") +
  scale_y_comma() +
  ggthemes::scale_color_tableau() +
  ggthemes::scale_fill_tableau() +
  labs(
    x=NULL, title="Weighted ILI by HHS Region (2013/4-2017/8)",
    caption="Source: CDC/#rstats cdcfluview"
  ) +
  facet_wrap(~region, scales="free_x", nrow=2) +
  theme(axis.text.x=element_text(size=9)) +
  theme(legend.position="none")
```


Weighted ILI by HHS Region (2013/4-2017/8)
Region 1
Region 2
Region 3
Region 4
Region 5
Region 6
Region 7
Region 8
Region 9
Region 10

```
library(statebins)

flu <- ili_weekly_activity_indicators(2017)

filter(flu, weekend == last(weekend)) %>%
  statebins(state_col = "statename",
            value_col = "activity_level",
            round = TRUE,
            ggplot2_scale_function =
                     viridis::scale_fill_viridis,
            name = "ILI Activity Level  ") +
  labs(title = sprintf(
    "U.S. ILI Weekly Activity : Week Ending %s / 2017-18
Season",
    last(flu$weekend))
  ) +
  theme_statebins(base_family = "Roboto Condensed")
```


U.S. ILI Weekly Activity : Week Ending 2018-01-13 / 2017-18 Season
ILI Activity Level
0.0 2.5 5.0 7.5 10.0

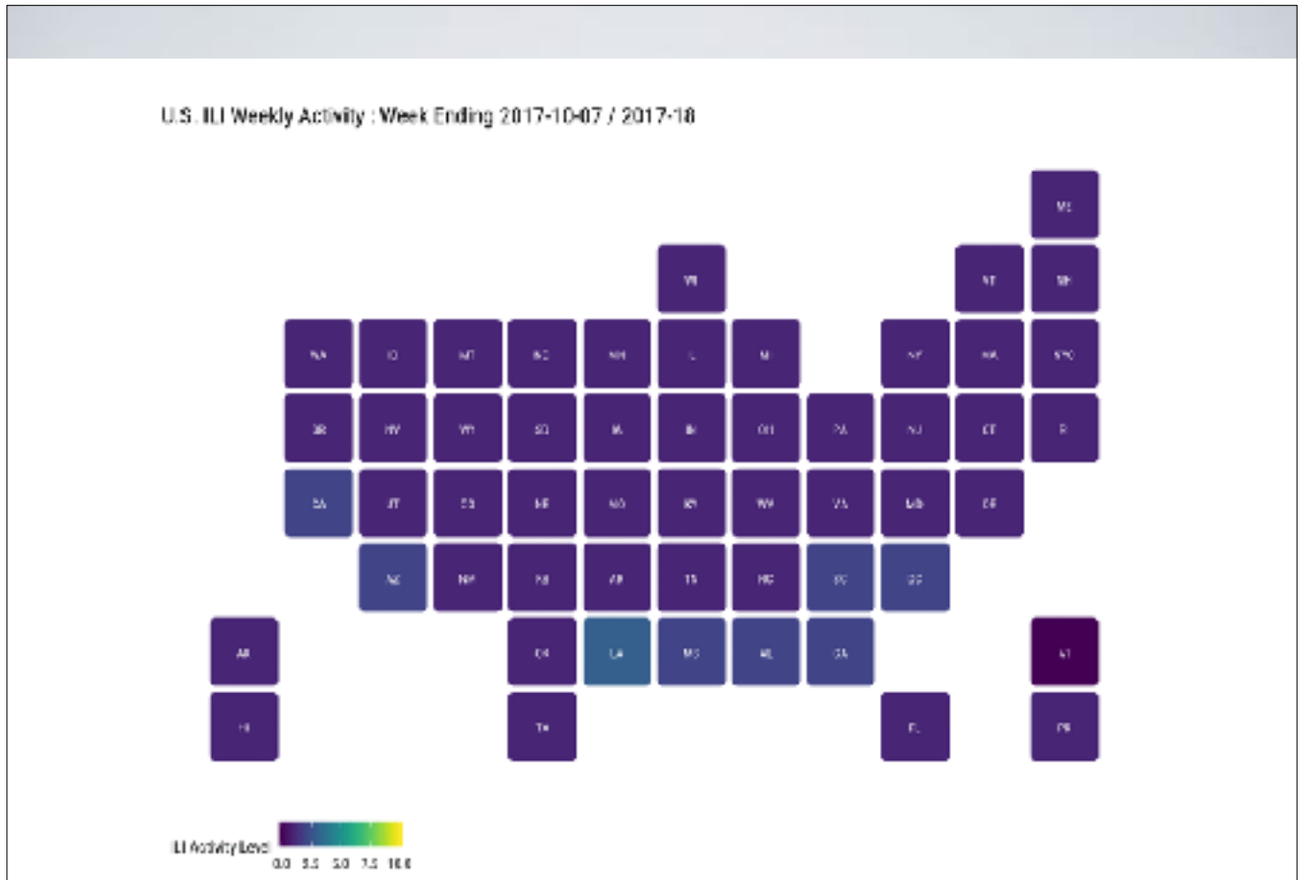
```r
library(statebins)
library(magick)
library(tidyverse)

flu <- ili_weekly_activity_indicators(2017)

frames <- image_graph(width=1800, height=1200, res=144)

arrange(flu, weekend) %>%
  pull(weekend) %>%
  unique() %>%
  map(~{
    filter(flu, weekend == .x) %>%
      statebins(state_col = "statename", value_col = "activity_level",
                round = TRUE,
                ggplot2_scale_function = viridis::scale_fill_viridis,
                limits=c(0,10),
                name = "ILI Activity Level  ") +
      labs(title = sprintf("U.S. ILI Weekly Activity : Week Ending %s / 2017-18", .x)) +
      theme_statebins(base_family = "Roboto Condensed") -> gg
    print(gg)
  }) -> y

gif <- image_animate(frames, 1)
image_write(gif, "fluview.gif")
```


U.S. ILI Weekly Activity : Week Ending 2017-10-07 / 2017-18
ILI Activity Level
0.0 2.5 5.0 7.5 10.0

age_group_distribution	Age Group Distribution of Influenza Positive Tests Reported by Public Health Laboratories
cdc_basemap	Retrieve CDC U.S. Basemaps
census_regions	Census Region Table
geographic_spread	State and Territorial Epidemiologists Reports of Geographic Spread of Influenza
get_weekly_flu_report	Retrieves (high-level) weekly (XML) influenza surveillance report from the CDC
hhs_regions	HHS Region Table
hospitalizations	Laboratory-Confirmed Influenza Hospitalizations
ilinet	Retrieve ILINet Surveillance Data
ili_weekly_activity_indicators	Retrieve weekly state-level ILI indicators per-state for a given season
mmwrid_map	MMWR ID to Calendar Mappings
mmwr_week	Convert a Date to an MMWR day+week+year
mmwr_weekday	Convert a Date to an MMWR weekday
mmwr_week_to_date	Convert an MMWR year+week or year+week+day to a Date object
pi_mortality	Pneumonia and Influenza Mortality Surveillance
state_data_providers	Retrieve metadata about U.S. State CDC Provider Data
surveillance_areas	Retrieve a list of valid sub-regions for each surveillance area.
who_nrevss	Retrieve WHO/NREVSS Surveillance Data

There are a few more functions and built-in data sources that map directly to the areas on the CDC FluView portal. If I can make any of the interfaces, documentation any better or add some functionality that would help you out, don't hesitate to file an issue on GitHub.

Why cdcfluview?

I mentioned the history of how FluView was created earlier. But, really. Why would I spend time creating this thing that requires care and feeding? And it most certainly does. Craig McGowan (a CDC researcher) notified me about the migration from Flash to something more open and safe last year and took a stab at ensuring all the functions worked after the cutover. I went back and reorganized the package interface to better map to the CDC's site and to also make the data a tad more useful. But, then again, why?

Initially, one of the drivers was to help someone not have to deal with the Flash interface. As a cybersecurity researcher, Flash is absolutely anathema. It's insecure, it destroys the semantic web and can be used to hide data. It's nothing but evil. A big motivator was to help ensure scientists and researchers could use a non-Flash-enabled browser. As you'll see in a bit, I'm a bit protective of the data science community as they're some of the most vulnerable users out there.

REPRODUCIBILITY

But, a bigger reason was reproducibility. What do I mean by reproducibility?

https://scholarlykitchen.sspnet.org/2017/05/24/reproducible-research-just-not-reproducible/

It means that someone can take your code and run it and get the same results (for the same data). Or be able to use it with new data but with consistent assumptions and diagnostic output. My own discipline is terrible about reproducible research, but then again, green text on black background blog posts count as research papers in cybersecurity, so it's no real surprise.

This is one area on the current CDC FluView portal. Folks are encouraged by the CDC to go there to get data they need.

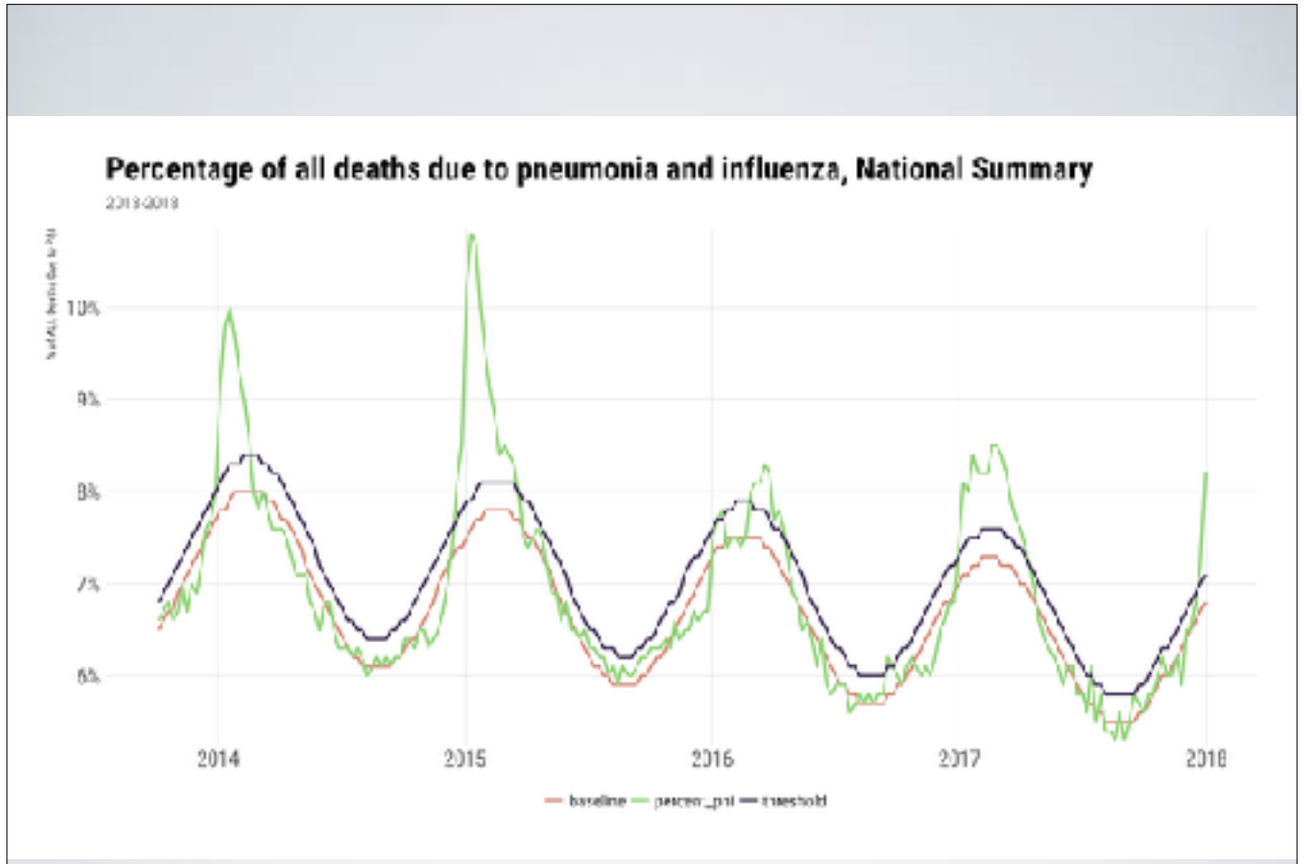
But, they were before the site was in HTML and the interface did change, so all the manual instructions for "go here and click X" were wrong and any automation based on instrumenting a browser also stopped working. Sure, my package had to be modified slightly, but the users had no real disruption in the API (I kept the old functionality even as I introduced a new, better one).

And, any workflow that involves a human means you can't really do:

```
mort <- pi_mortality(years=2013:2018)

select(mort, wk_end, baseline, threshold, percent_pni) %>%
  gather(measure, value, -wk_end) %>%
  ggplot(aes(wk_end, value, group=measure)) +
  geom_line(aes(color=measure)) +
  labs(
    x=NULL, y="% of ALL Deaths Due to P&I",
    title="Percentage of all deaths due to pneumonia and influenza, National Summary",
    subtitle="2013-2018"
  ) +
  theme_ipsum_rc(grid="XY") +
  scale_color_ipsum(name=NULL) +
  scale_y_percent() +
  theme(legend.position="bottom") +
  theme(legend.direction="horizontal")
```

this. And run it every week to get



this. It's not reproducible. And if you manually download the file to the wrong place, things just stop working. I could go on for a while abt reproducibility, but let's keep moving.

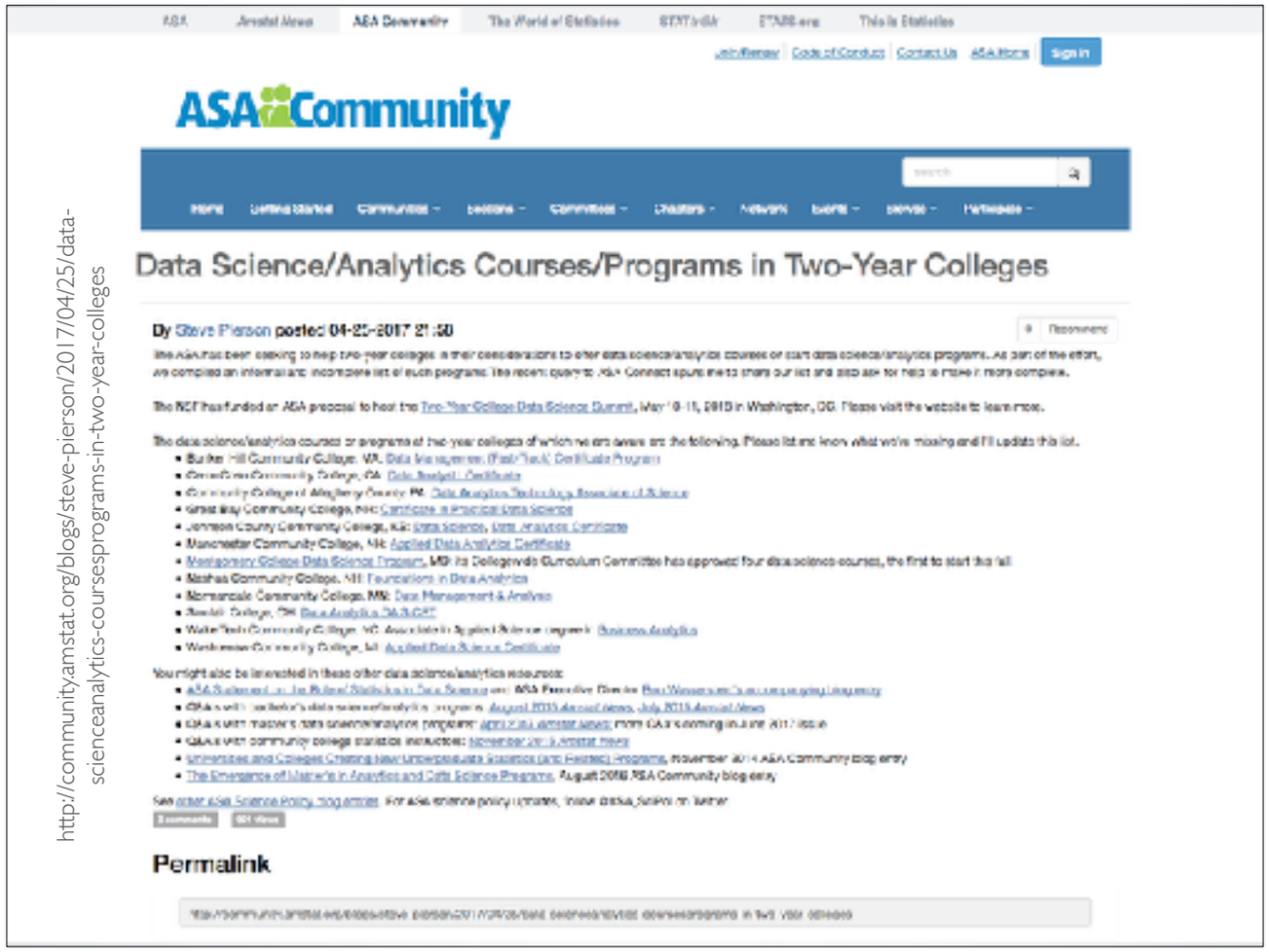
ENABLING CITIZEN DATA SCIENCE

Another "cause" / driver for me is to ensure there's data literacy in the general public and enable+enable "Citizen Data Science"

We're producing scads of data in virtually every field and publishing it, but a good chunk of the general public has no idea how to consume it or work with it. I and my students (I teach data science in community colleges … more on that in a bit, too) knew the flu season was getting worse before it made headlines … because we were looking at the data. I've got my own personal dashboard and this package is part of it (partly so I can monitor for changes in the CDC's hidden API).

We really need to find ways of getting the public more interested in this type of data and other data if we're going to have an informed society who can put checks and balances on things and also not just take the word of the media (or, even government) for things.

With the current anti-data/anti-science makeup of the U.S. government, more and more data sources are going away, so having interfaces like cdcfluview also helps in archiving this data. But, I'm almost on a soapbox and shld back away slowly and move on.



http://community.amstat.org/blogs/steve-pierson/2017/04/25/data-scienceanalytics-coursesprograms-in-two-year-colleges

I helped develop a data science program for the college system of New Hampshire and am involved with other efforts at the ASA on developing a core, standard curriculum for 2 year colleges to get students through either a certificate program in data science or a full associates degree. This is designed so that the average citizen can come in and leave with data literacy and data science skills and also to to give 4-year and beyond students a leg up when they take their biology, ecology, sociology or other major courses. A friendly interface to CDC surveillance data truly makes the topic far more approachable.

Why **R** for ~~Disease-surveillance~~ **Everything**?!

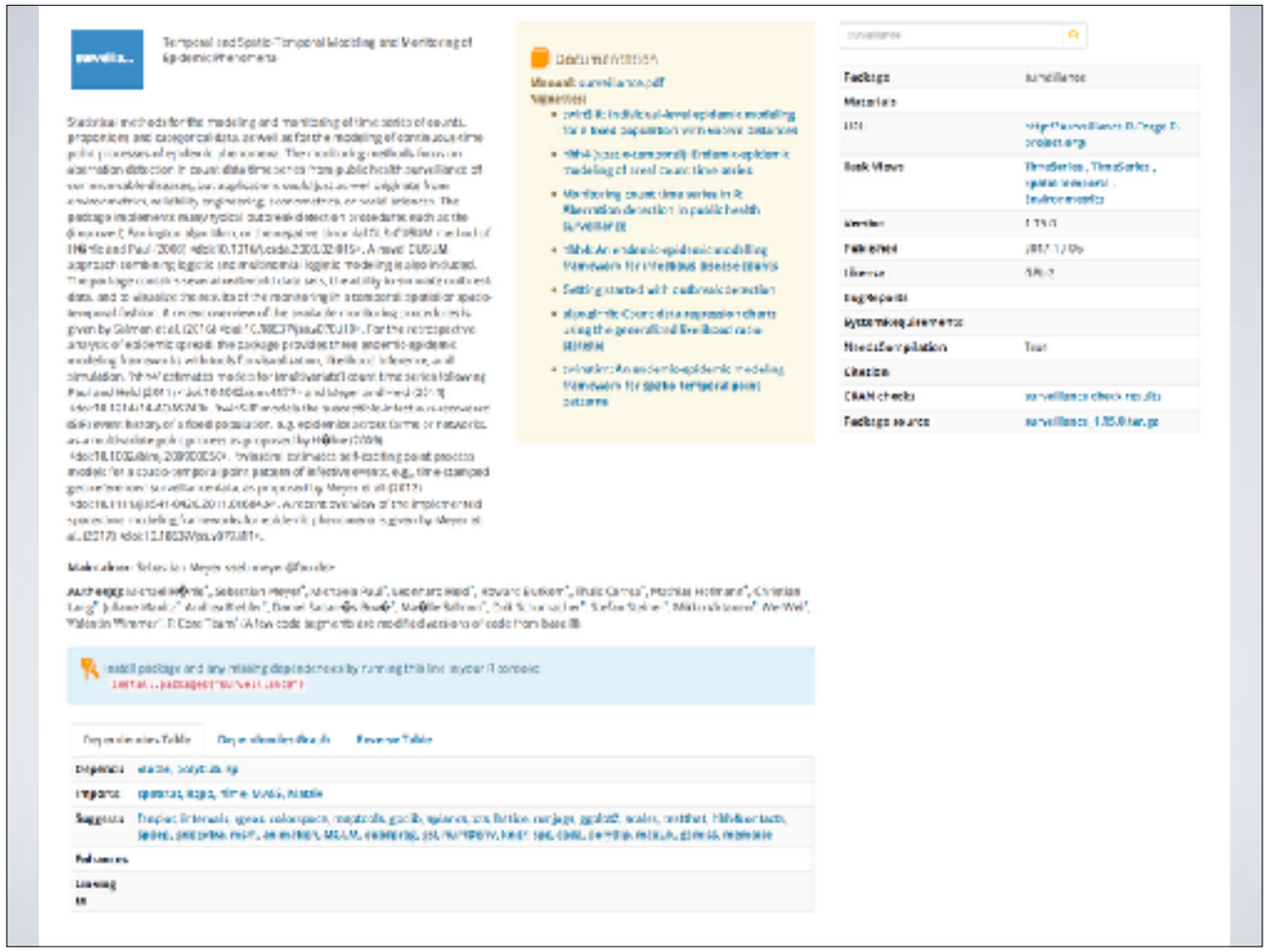
The package and the courses I teach are all in R. But why R?


Deep
down you
already
know the
truth.

Packages are one big area that make R great for data science. Sure, Python has orders of magnitude more overall packages, but R's packages are usually written by practitioners who know what others need and try to ensure the focus is on data quality and also work to make them easy to use.

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4050851/

You've got many R packages for disease surveillance.



Some don't even try to have clever names (many R packages try to be clever name-wise)

CONNECTIVITY

- Spark
- Apache Drill
- Every database (without needing to know SQL)
- Every file format

But, there's also connectivity.

R was meant to be extensible and you can call python from it or use any (and I mean any) database or data format. It has support and with a friendly interface.

VISUALIZATIONS

- ggplot2
- htmlwidgets (**plotly**)
- Shiny
- (ugh) Tableau
- Many others

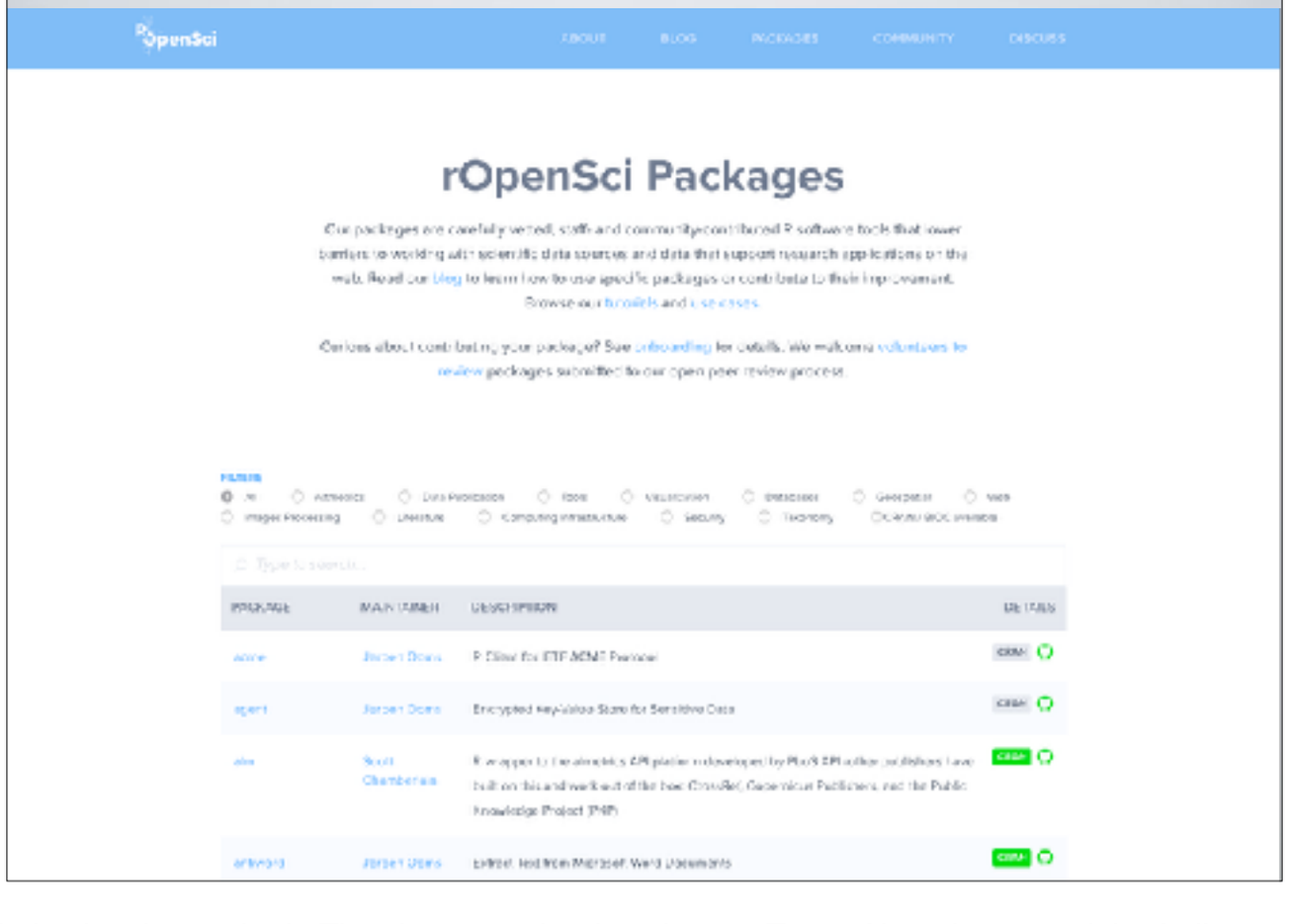
I tend to overemphasize the vis side of data science. One reason for that is that cybersecurity folks don't trust statistics and are still pretty wary about machine learning. We're still in the "counting things" phase (it's really a protoscience). All of the visualizations I showed today were in ggplot2. It's by far the best static vis system. Period.

But, there's a dynamic side to R and htmlwidgets enable easy creation of shareable interactive visualizations.

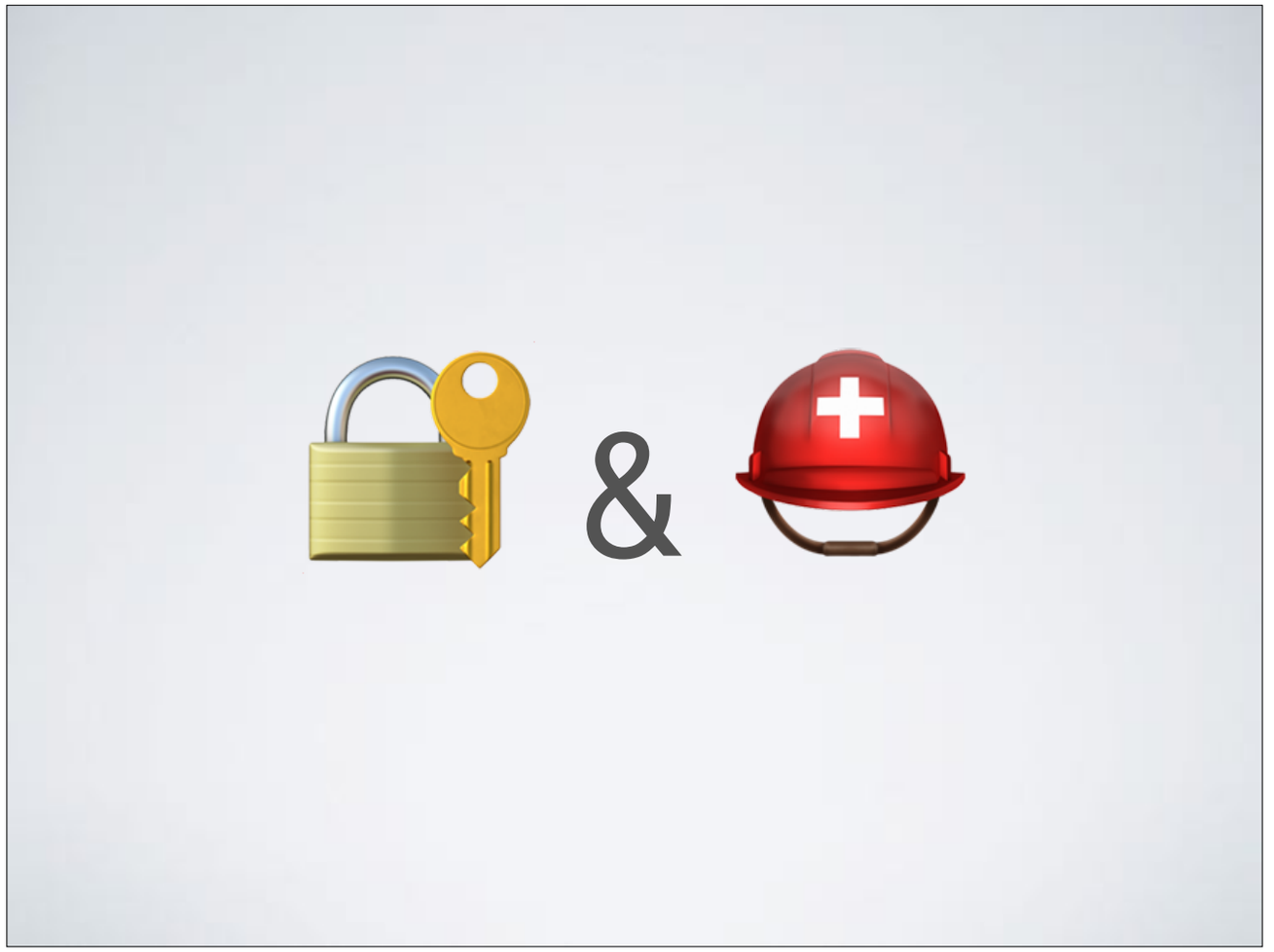
Shiny is a web app framework for R that makes it straightforward (I say "easy" deliberately and I mean straightforward here…it's got a learning curve) to create high quality apps backed by your amazing predictive models.

Likely the best reason to consider R for anything (but especially for disease surveillance) is the community. I work with many of the rOpenSci folks and they're amazing.

https://ropensci.org/packages/



If you don't know about rOpenSci, definitely give them a look. They have scads of packages. Tons of learning materials. A discussion forum and more. They are the smartest and truly nicest folks I know.

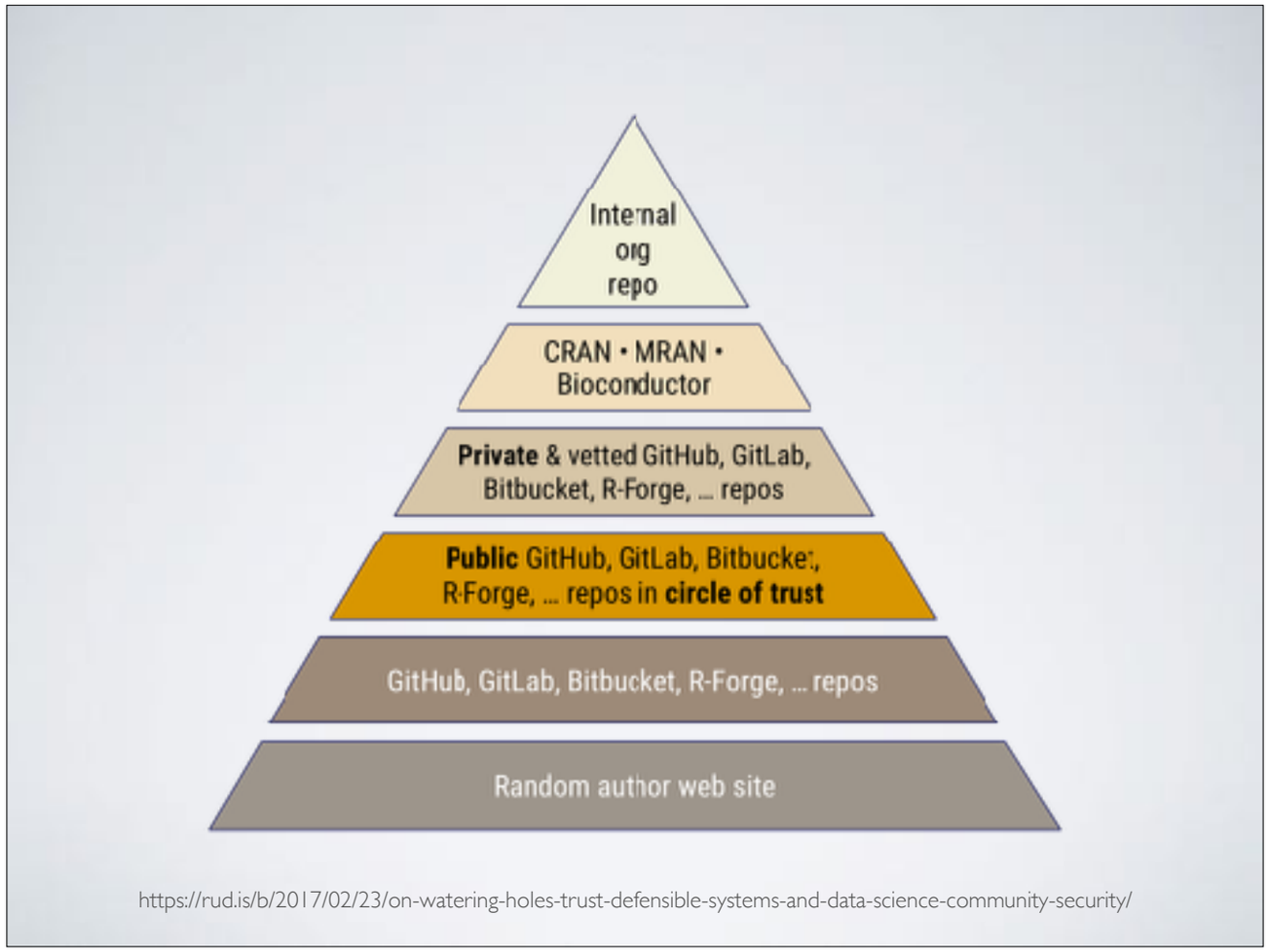


Time's closing in, but I can't leave you without talking about security and safety when it comes to data science.

When you download an app, package or even visit a link on your workstation, you're opening up you and your computer to harm.

I'm not suggesting you go live in a cave in Maine (though it is nice up here!)

But, you should be aware of the potential safety & security issues whenever you decided to load up some new "data science" thing.



https://rud.is/b/2017/02/23/on-watering-holes-trust-defensible-systems-and-data-science-community-security/

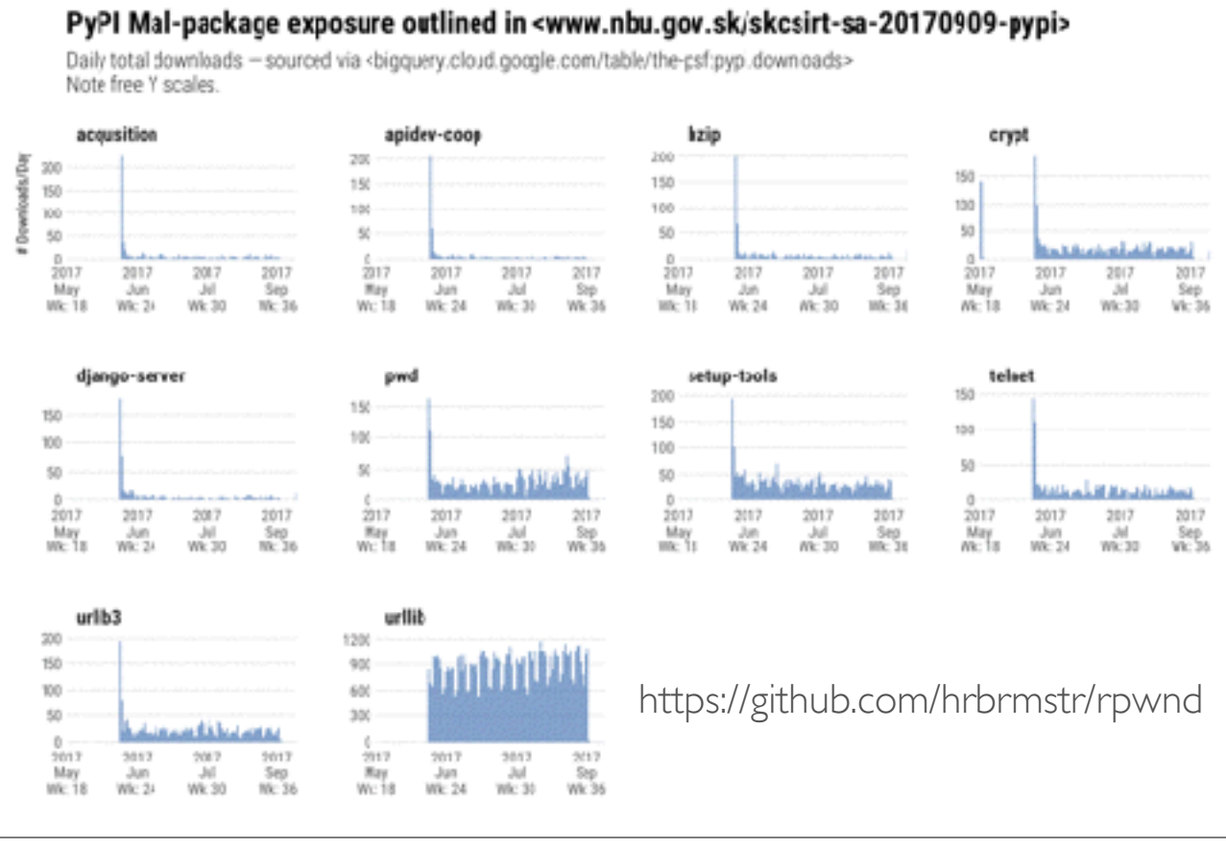
I put a blog post together last year that talks about this in detail, so I'll just hit the highlights. First, you really need to be careful about what you decided to load up in R (or Python, etc). This is my trust pyramid for packages. Stuff at work or home is super gd. CRAN & MRAN (which is fed from CRAN) are good too. I should really remove Bioconductor from that row, though. It's a hot mess.

If I know the folks, private/vetted repos are ok as well.

Things on github that are by contributors I know and especially signed (ask me about that if you want to know more) are ok, too, but I'm very wary of anything outside my circle of trust and read through every line before using (even the C/C++ code).

Generic GitHub repos or random code on web sites are right out unless you read every line and know what you're doing.

https://rud.is/b/2017/09/15/its-a-fake-%F0%9F%93%A6-revisiting-trust-in-foss-ecosystems/
https://arstechnica.com/information-technology/2017/09/devs-unknowingly-use-malicious-modules-put-into-official-python-repository/



Think it can't happen? The Python packages listed here were all "hacked" last year. The graphs show (free Y scales) the number of weekly downloads of them. Sure, these aren't "data sci" packages (well the URL ones could be used by some) but these are the known ones. PyPI security & integrity is laughable, especially when compared to CRAN.

What could a malicious package do? Hit the URL at the bottom to see more (and, trying it out will have a bigger impact than if I just tell you).

I could go into much more on this, but the core message is that you are a target. Whether it's organized crime, nation-states (even your own) or some rogue organization, they all know they can collect or corrupt data to cause whatever end they do. So, while you also need to worry about data quality, you also need to be concerned with what you click, what you use and how much trust you're putting in those things.

- bob@rud.is
- bob_rudis@rapid7.com
- @hrbrmstr
- https://rud.is/b/
- https://github.com/hrbrmstr/
- https://github.com/hrbrmstr/2018-01-24-isds

I've covered a wide array of topics and this presentation and the R code in it + all the links are in the github repo at the bottom.

If you're on Twitter, I do regular "PSAs" about important security topics you need to know about (like the latest vulns/attacks and how to protect yourself).

But, don't think twice about e-mailing or @'ing me. I'd love to help you be more secure or help ensure I make things that are useful to you.

With that I thank you for your time and have just a few minutes left for more questions.