Analysis of Programming Languages

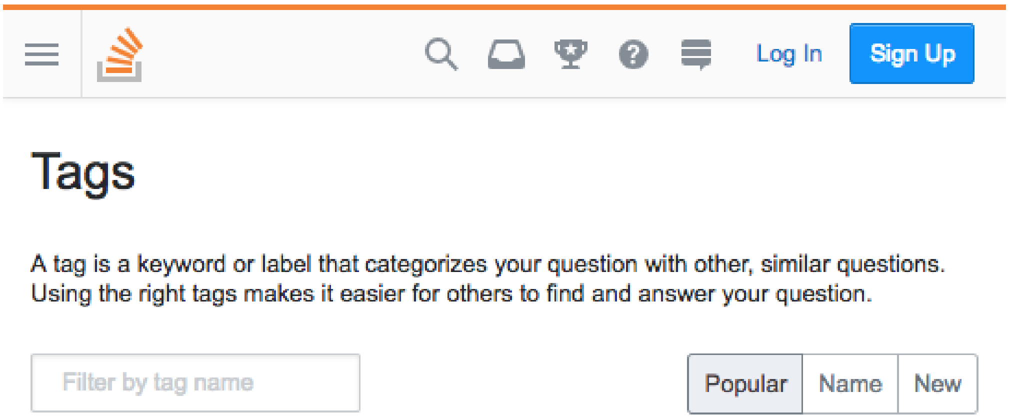
- **by Madhura Ashtekar**

1. Data on tags over time

How can we tell what programming languages and technologies are used by the most people? How about what languages are growing and which are shrinking, so that we can tell which are most worth investing time in?

One excellent source of data is Stack Overflow, a programming question and answer site with more than 16 million questions on programming topics. By measuring the number of questions about each technology, we can get an approximate sense of how many people are using it. We're going to use open data from the Stack Exchange Data Explorer to examine the relative popularity of languages like R, Python, Java and Javascript have changed over time.

Each Stack Overflow question has a `tag`, which marks a question to describe its topic or technology. For instance, there's a tag for languages like R or Python, and for packages like ggplot2 or pandas.



Tags

A tag is a keyword or label that categorizes your question with other, similar questions.
Using the right tags makes it easier for others to find and answer your question.

We'll be working with a dataset with one observation for each tag in each year. The dataset includes both the number of questions asked in that tag in that year, and the total number of questions asked in that year.

In [ ]:

```
# Load libraries
library(readr)
library(dplyr)

# Load dataset
by_tag_year <- read_csv('datasets/by_tag_year.csv')

# Inspect the dataset
by_tag_year
```

```
Parsed with column specification:
cols(
  year = col_integer(),
  tag = col_character(),
  number = col_integer(),
  year_total = col_integer()
)
```

year	tag	number	year_total
2008	.htaccess	54	58390
2008	.net	5910	58390
2008	.net-2.0	289	58390
2008	.net-3.5	319	58390

year	tag	number	year_total
2008	.net-4.0	6	58390
2008	.net-assembly	3	58390
2008	.net-core	1	58390
2008	2d	42	58390
2008	32-bit	19	58390
2008	32bit-64bit	4	58390
2008	3d	73	58390
2008	64bit	149	58390
2008	abap	10	58390
2008	absolute	1	58390
2008	abstract	5	58390
2008	abstract-class	27	58390
2008	abstract-syntax-tree	6	58390
2008	accelerometer	3	58390
2008	access	1	58390
2008	access-control	12	58390
2008	accessibility	26	58390
2008	access-vba	50	58390
2008	access-violation	4	58390
2008	accordion	9	58390
2008	acl	11	58390
2008	acrobat	10	58390
2008	action	10	58390
2008	actionlistener	4	58390
2008	actionmailer	3	58390
2008	actionscript	136	58390
...	...	...	...
2018	yaml	648	1085170
2018	yarn	357	1085170
2018	yeoman	36	1085170
2018	yesod	41	1085170
2018	yield	69	1085170
2018	yii	269	1085170
2018	yii2	1181	1085170
2018	yii2-advanced-app	209	1085170
2018	yocto	288	1085170
2018	youtube	676	1085170
2018	youtube-api	473	1085170
2018	youtube-api-v3	223	1085170
2018	youtube-data-api	203	1085170
2018	yui	5	1085170
2018	yum	98	1085170
2018	z3	124	1085170
2018	zend-db	11	1085170
2018	zend-form	13	1085170

year	tag	number	year_total
2018	zend-framework	188	1085170
2018	zend-framework2	108	1085170
2018	zeromq	168	1085170
2018	z-index	107	1085170
2018	zip	410	1085170
2018	zipfile	115	1085170
2018	zk	35	1085170
2018	zlib	89	1085170
2018	zoom	196	1085170
2018	zsh	175	1085170
2018	zurb-foundation	182	1085170
2018	zxing	95	1085170

2. Now in fraction format

This data has one observation for each pair of a tag and a year, showing the number of questions asked in that tag in that year and the total number of questions asked in that year. For instance, there were 54 questions asked about the `.htaccess` tag in 2008, out of a total of 58390 questions in that year.

Rather than just the counts, we're probably interested in a percentage: the fraction of questions that year that have that tag. So let's add that to the table.

In [ ]:

```
# Add fraction column
by_tag_year_fraction <- by_tag_year %>%
    mutate(fraction = number / year_total)

# Print the new table
by_tag_year_fraction
```

year	tag	number	year_total	fraction
2008	.htaccess	54	58390	9.248159e-04
2008	.net	5910	58390	1.012160e-01
2008	.net-2.0	289	58390	4.949478e-03
2008	.net-3.5	319	58390	5.463264e-03
2008	.net-4.0	6	58390	1.027573e-04
2008	.net-assembly	3	58390	5.137866e-05
2008	.net-core	1	58390	1.712622e-05
2008	2d	42	58390	7.193013e-04
2008	32-bit	19	58390	3.253982e-04
2008	32bit-64bit	4	58390	6.850488e-05
2008	3d	73	58390	1.250214e-03
2008	64bit	149	58390	2.551807e-03
2008	abap	10	58390	1.712622e-04
2008	absolute	1	58390	1.712622e-05
2008	abstract	5	58390	8.563110e-05
2008	abstract-class	27	58390	4.624079e-04
2008	abstract-syntax-tree	6	58390	1.027573e-04
2008	accelerometer	3	58390	5.137866e-05

year	tag	number	year_total	fraction
2008	access	1	58390	1.712622e-05
2008	access-control	12	58390	2.055146e-04
2008	accessibility	26	58390	4.452817e-04
2008	access-vba	50	58390	8.563110e-04
2008	access-violation	4	58390	6.850488e-05
2008	accordion	9	58390	1.541360e-04
2008	acl	11	58390	1.883884e-04
2008	acrobat	10	58390	1.712622e-04
2008	action	10	58390	1.712622e-04
2008	actionlistener	4	58390	6.850488e-05
2008	actionmailer	3	58390	5.137866e-05
2008	actionscript	136	58390	2.329166e-03
...	...	...	...	...
2018	yaml	648	1085170	5.971415e-04
2018	yarn	357	1085170	3.289807e-04
2018	yeoman	36	1085170	3.317453e-05
2018	yesod	41	1085170	3.778210e-05
2018	yield	69	1085170	6.358451e-05
2018	yii	269	1085170	2.478874e-04
2018	yii2	1181	1085170	1.088309e-03
2018	yii2-advanced-app	209	1085170	1.925966e-04
2018	yocto	288	1085170	2.653962e-04
2018	youtube	676	1085170	6.229439e-04
2018	youtube-api	473	1085170	4.358764e-04
2018	youtube-api-v3	223	1085170	2.054978e-04
2018	youtube-data-api	203	1085170	1.870675e-04
2018	yui	5	1085170	4.607573e-06
2018	yum	98	1085170	9.030843e-05
2018	z3	124	1085170	1.142678e-04
2018	zend-db	11	1085170	1.013666e-05
2018	zend-form	13	1085170	1.197969e-05
2018	zend-framework	188	1085170	1.732447e-04
2018	zend-framework2	108	1085170	9.952358e-05
2018	zeromq	168	1085170	1.548145e-04
2018	z-index	107	1085170	9.860206e-05
2018	zip	410	1085170	3.778210e-04
2018	zipfile	115	1085170	1.059742e-04
2018	zk	35	1085170	3.225301e-05
2018	zlib	89	1085170	8.201480e-05
2018	zoom	196	1085170	1.806169e-04
2018	zsh	175	1085170	1.612651e-04
2018	zurb-foundation	182	1085170	1.677157e-04
2018	zxing	95	1085170	8.754389e-05

3. Has R been growing or shrinking?

So far we've been learning and using the R programming language. Wouldn't we like to be sure it's a good investment for the future? Has it been keeping pace with other languages, or have people been switching out of it?

Let's look at whether the fraction of Stack Overflow questions that are about R has been increasing or decreasing over time.

In [ ]:

```
# Filter for R tags
r_over_time <- by_tag_year_fraction %>%
    filter(tag == 'r')

# Print the new table
r_over_time
```

year	tag	number	year_total	fraction
2008	r	8	58390	0.0001370098
2009	r	524	343868	0.0015238405
2010	r	2270	694391	0.0032690516
2011	r	5845	1200551	0.0048685978
2012	r	12221	1645404	0.0074273552
2013	r	22329	2060473	0.0108368321
2014	r	31011	2164701	0.0143257660
2015	r	40844	2219527	0.0184021190
2016	r	44611	2226072	0.0200402323
2017	r	54415	2305207	0.0236052554
2018	r	28938	1085170	0.0266667895

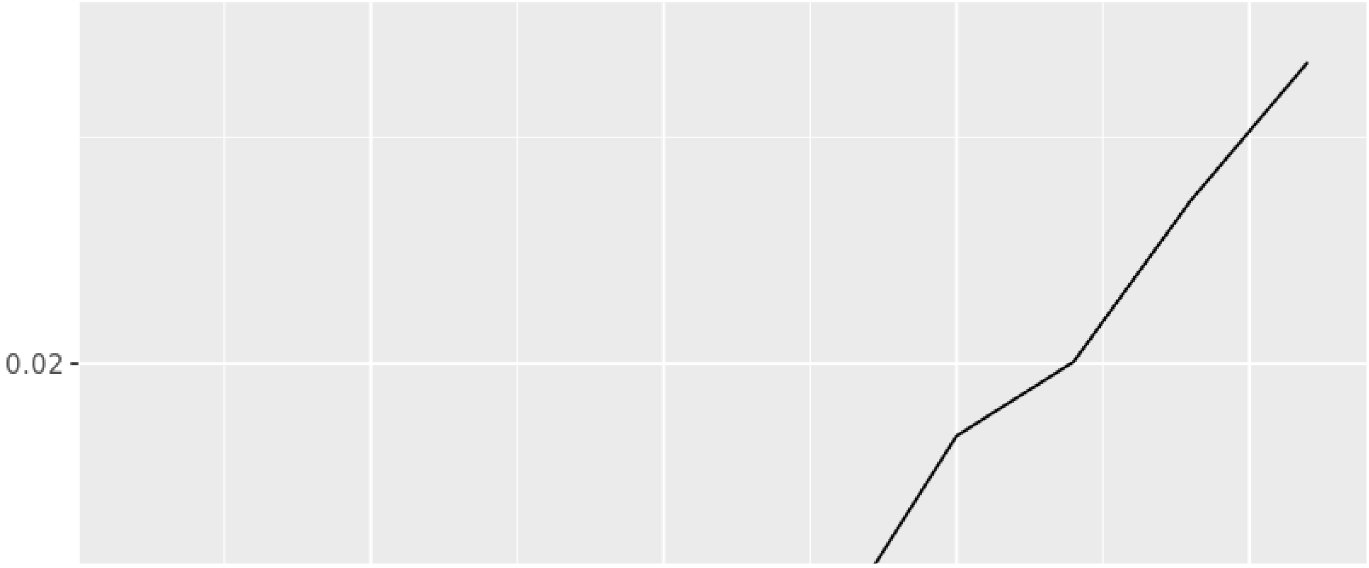
4. Visualizing change over time

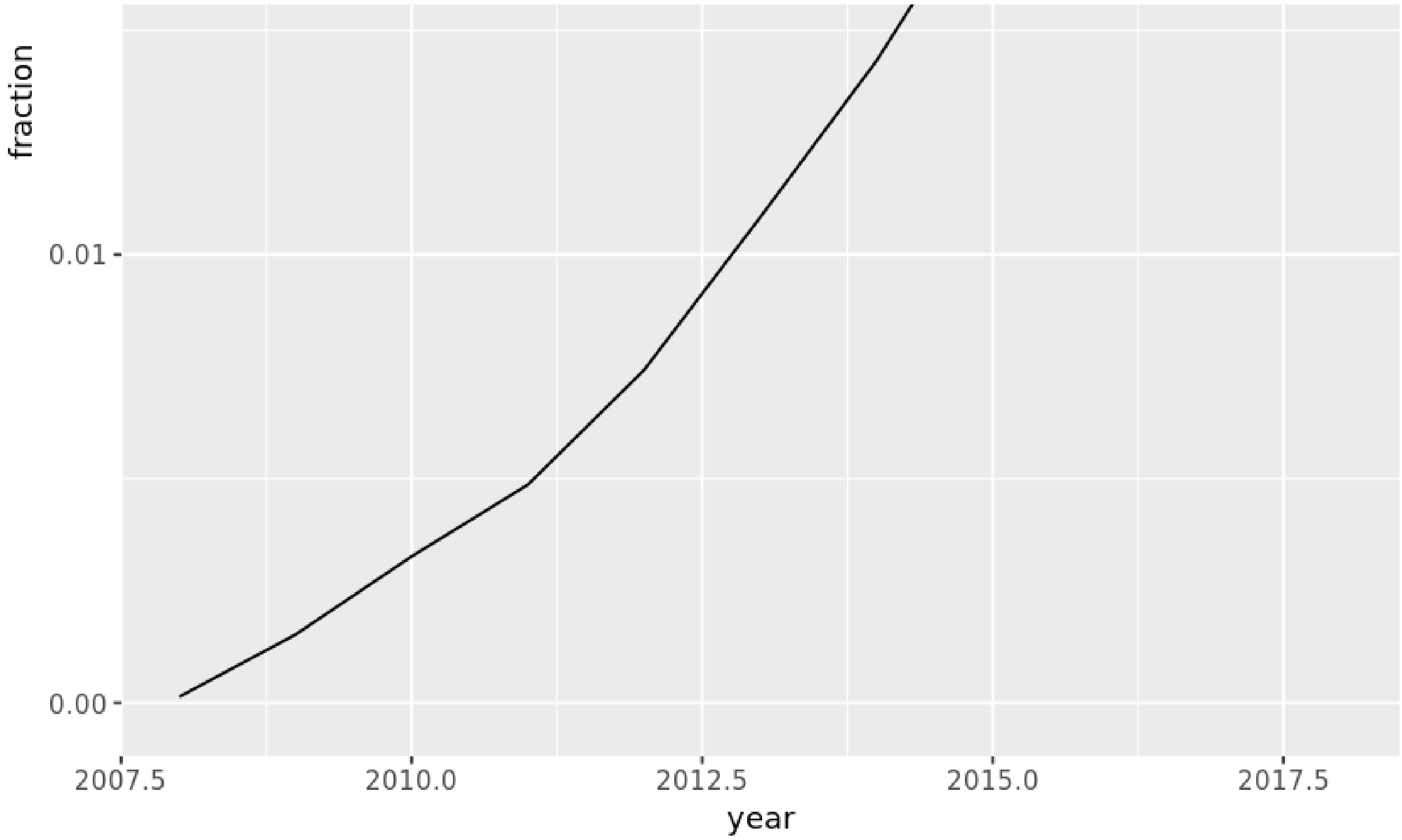
Rather than looking at the results in a table, we often want to create a visualization. Change over time is usually visualized with a line plot.

In [ ]:

```
# Load ggplot2
library(ggplot2)

# Create a line plot of fraction over time
ggplot(r_over_time, aes(x=year, y=fraction)) +
    geom_line()
```





5. How about dplyr and ggplot2?

Based on that graph, it looks like R has been growing pretty fast in the last decade. Good thing we're practicing it now!

Besides R, two other interesting tags are dplyr and ggplot2, which we've already used in this analysis. They both also have Stack Overflow tags!

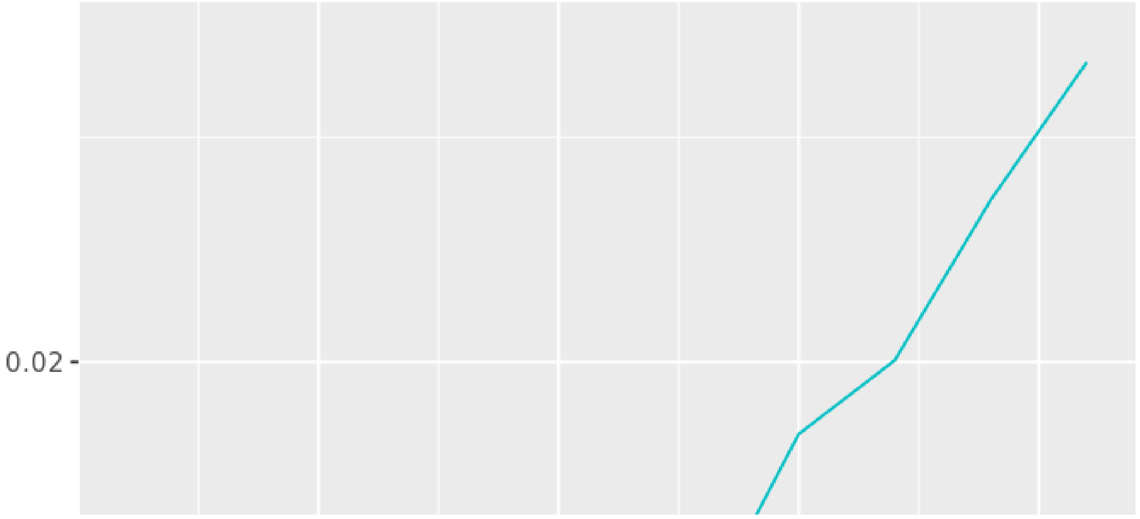
Instead of just looking at R, let's look at all three tags and their change over time. Are each of those tags increasing as a fraction of overall questions? Are any of them decreasing?

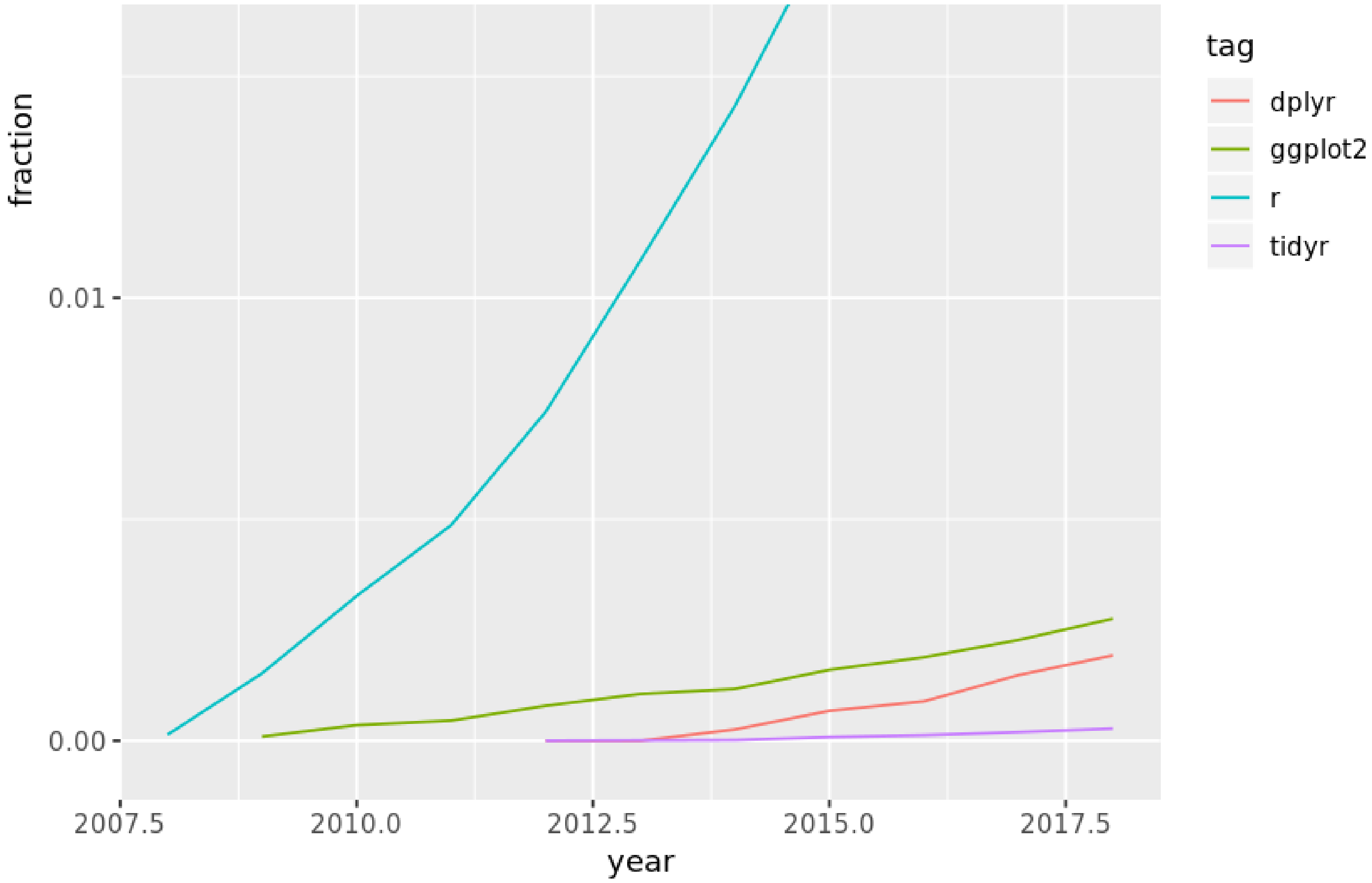
In [ ]:

```
# A vector of selected tags
selected_tags <- c('r', 'dplyr', 'ggplot2','tidyr')

# Filter for those tags
selected_tags_over_time <- by_tag_year_fraction %>%
    filter(tag %in% selected_tags)

# Plot tags over time on a line plot using color to represent tag
ggplot(selected_tags_over_time, aes(x=year, y=fraction, color=tag)) +
    geom_line()
```





6. What are the most asked-about tags?

It's sure been fun to visualize and compare tags over time. The dplyr and ggplot2 tags may not have as many questions as R, but we can tell they're both growing quickly as well.

We might like to know which tags have the most questions *overall*, not just within a particular year. Right now, we have several rows for every tag, but we'll be combining them into one. That means we want `group_by()` and `summarize()` .

Let's look at tags that have the most questions in history.

In [ ]:

```
# Find total number of questions for each tag
sorted_tags <- by_tag_year %>%
group_by(tag) %>% summarize(tag_total=n()) %>% arrange(-tag_total)

# Print the new table
print(sorted_tags)
```

```
# A tibble: 4,080 x 2
   tag            tag_total
   <chr>              <int>
 1 .htaccess             11
 2 .net                  11
 3 .net-2.0              11
 4 .net-3.5              11
 5 .net-4.0              11
 6 .net-assembly         11
 7 2d                    11
 8 32-bit                11
 9 32bit-64bit           11
10 3d                    11
# ... with 4,070 more rows
```

7. How have large programming languages changed over time?

We've looked at selected tags like R, ggplot2, and dplyr, and seen that they're each growing. What tags might

We've looked at selected tags like R, ggplot2, and dplyr, and seen that they're each growing. What tags might be *shrinking*? A good place to start is to plot the tags that we just saw that were the most-asked about of all time, including JavaScript, Java and C#.

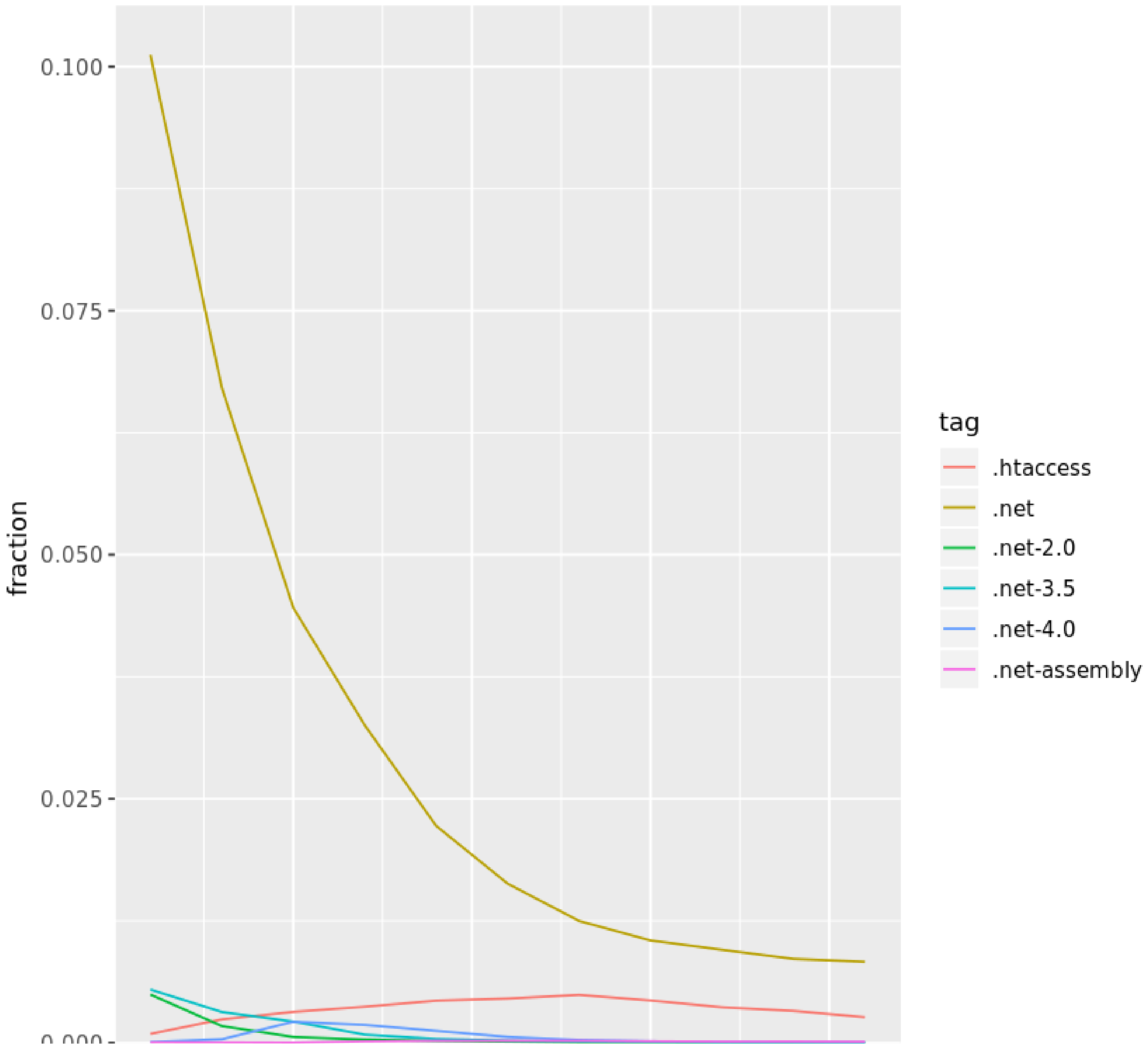
In [ ]:

```
# Get the six largest tags
highest_tags <- head(sorted_tags$tag)

# Filter for the six largest tags
by_tag_subset <- filter(by_tag_year_fraction, tag %in% highest_tags)
print(by_tag_subset)

# Plot tags over time on a line plot using color to represent tag
ggplot(by_tag_subset, aes(x = year, y = fraction, col = tag)) + geom_line()
```

```
# A tibble: 66 x 5
    year tag            number year_total  fraction
   <int> <chr>           <int>      <int>     <dbl>
 1  2008 .htaccess          54      58390 0.000925
 2  2008 .net             5910      58390 0.101
 3  2008 .net-2.0          289      58390 0.00495
 4  2008 .net-3.5          319      58390 0.00546
 5  2008 .net-4.0            6      58390 0.000103
 6  2008 .net-assembly       3      58390 0.0000514
 7  2009 .htaccess         828     343868 0.00241
 8  2009 .net            23076     343868 0.0671
 9  2009 .net-2.0          593     343868 0.00172
10  2009 .net-3.5         1087     343868 0.00316
# ... with 56 more rows
```



8. Some more tags!

Wow, based on that graph we've seen a lot of changes in what programming languages are most asked about. C# gets fewer questions than it used to, and Python has grown quite impressively.

This Stack Overflow data is incredibly versatile. We can analyze *any* programming language, web framework, or tool where we'd like to see their change over time. Combined with the reproducibility of R and its libraries, we have ourselves a powerful method of uncovering insights about technology.

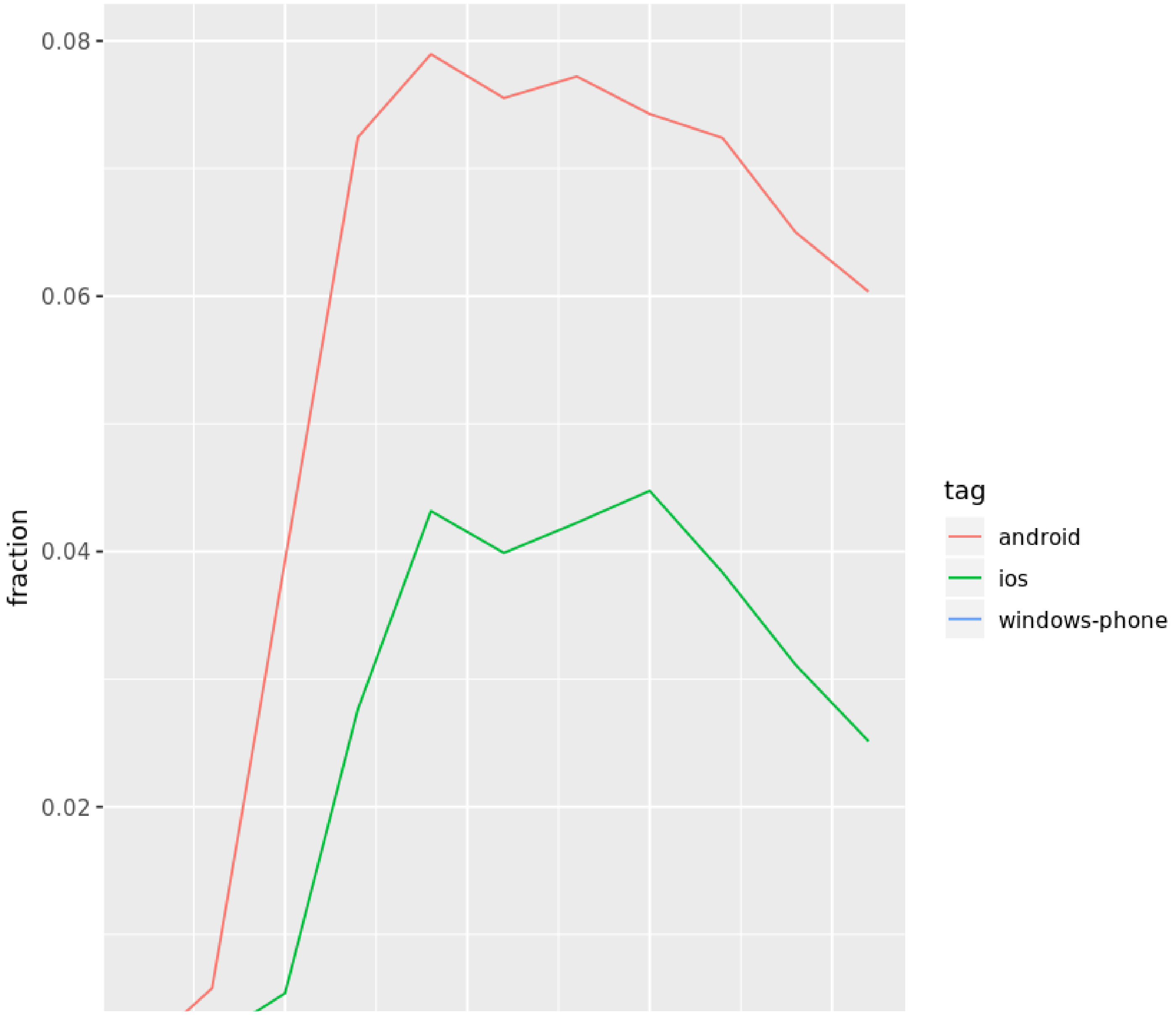
To demonstrate its versatility, let's check out how three big mobile operating systems (Android, iOS, and Windows Phone) have compared in popularity over time. But remember: this code can be modified simply by changing the tag names!

In [ ]:

```
# Get tags of interest
my_tags <- c("android", "ios", "windows-phone")

# Filter for those tags
by_tag_subset <- filter(by_tag_year_fraction, tag %in% my_tags)

# Plot tags over time on a line plot using color to represent tag
ggplot(by_tag_subset, aes(x = year, y = fraction, col = tag)) + geom_line()
```



0.00
2007.5
2010.0
2012.5
2015.0
2017.5
year