```
knitr::opts_chunk$set(  echo=TRUE, eval=FALSE)
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-26. For overview type 'help("mgcv-package")'.
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.7
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
```

```
library(brms);library(ggplot2);library(scales)
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.7.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
## Run theme_set(theme_default()) to use the default bayesplot theme.
```

```
##
## Attaching package: 'brms'
```

```
## The following objects are masked from 'package:mgcv':
##
##     s, t2
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(directlabels)
```

```
##
## Attaching package: 'directlabels'
```

```
## The following object is masked from 'package:nlme':
##
##     gapply
```

Estimating Desktop Browser Market Size

Desktop Firefox has been losing users for the past few years and we have also been losing market share. We've generally attributed this to a combination of Chrome taking our users and the generally declining state of the desktop market.

We can use the combination of our user numbers (MAU), which we know, and an external estimate of our market share to estimate the overall size of the market. We have MAU values for each day of the month, but market share only for a month as a whole. As an approximation, we take the MAU value for the last day of the month and attribute it to the month as a whole.

TECHNICAL NOTE: In the current version of this code, we actually assign all these values to the first day of the month rather than the last, which means that all the graphs are actually off by a month or so. This doesn't affect the general trend, but I need to write some code to fix it.

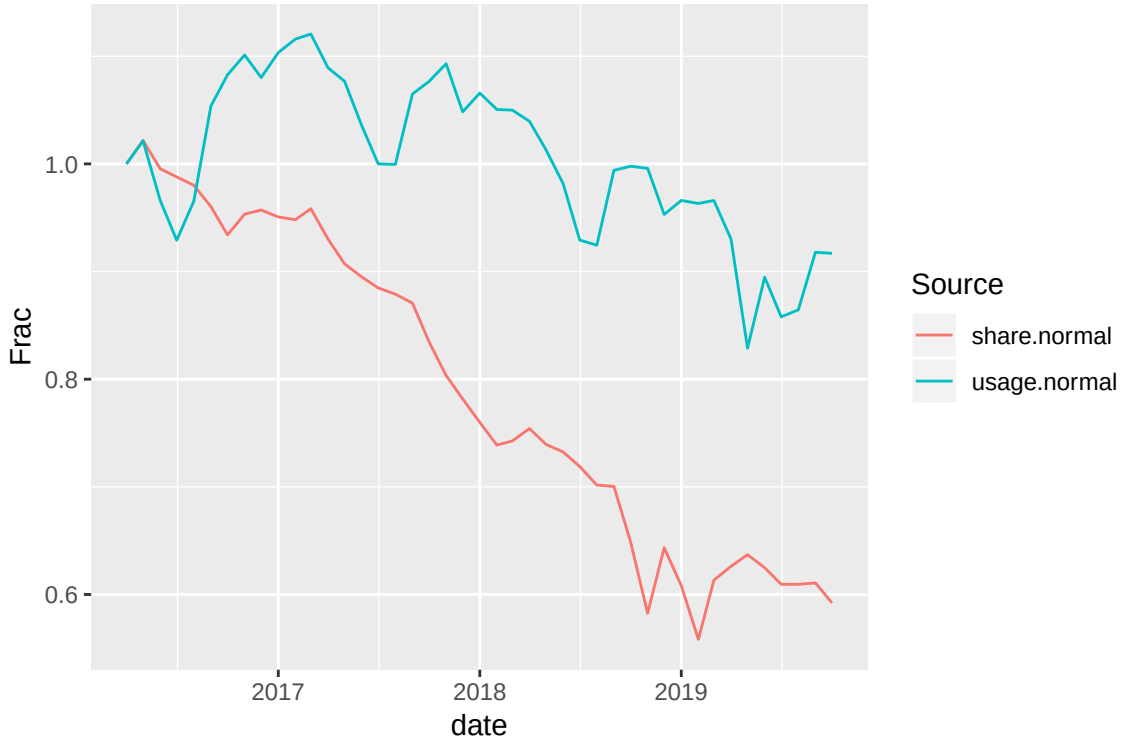
```
ms.ww <- fread("browser-ww-monthly-200901-201910-first.csv")[,date:=as.Date(Date)]
ms.us <- fread("browser-us-monthly-200901-201910-first.csv")[,date:=as.Date(Date)]
mau <- fread("Desktop_MAU_in_US_and_ROW_2019_11_22-last.csv")[,date := as.Date(date)]
adi <- fread("adi-mean.csv")[,date := as.Date(date)]
nm.ww <- fread("netmarketshare.csv")[,date:=as.Date(date)]

# Trim these to the same dates
# First ADI
ms.adi <- ms.ww[date >= min(adi$date) & date <= max(adi$date)]
df.adi <- data.table(date = ms.adi$date, usage = adi$adi, share = ms.adi$Firefox/100)

# Then MAU for StatCounter
ms.ww <- ms.ww[date >= min(mau$date)]
ms.us <- ms.us[date >= min(mau$date)]
mau.st <- mau[date <= max(ms.ww$date)]
df.ww <- data.table(date = ms.ww$date, usage = mau.st$mau_row + mau.st$mau_us, share = ms.ww$Firefox/100)
df.us <- data.table(date = ms.us$date, usage = mau.st$mau_us, share = ms.us$Firefox/100)

# Finally MAU for NetMarketShare
mau.nm <- mau[date >= min(nm.ww$date)]
df.nm <- data.table(date = nm.ww$date, usage = mau.nm$mau_row + mau.nm$mau_us, share = nm.ww$Firefox/100)
```

Worldwide Market Size

Fhe following figure shows the decline of MAU and market share from the period of 2016-04-01 to 2019-10-01. The MAU data comes from our own telemetry and the market share data comes from StatCounter. Each of these is normalized to the start of the period. Note that market share is declining faster than MAU.

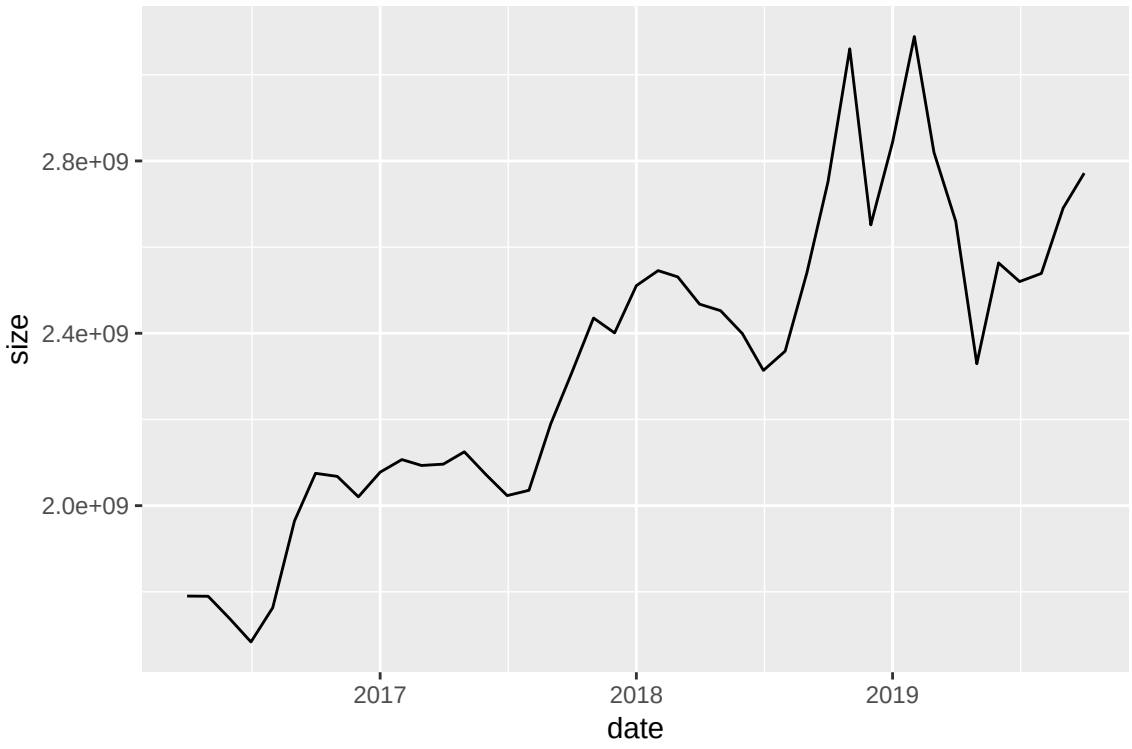
```
df <- df.ww

df$size <- df$usage/df$share
df$usage.normal <- df$usage/df$usage[1]
df$share.normal <- df$share/df$share[1]
df.long <- gather(df, key="Source", value="Frac", usage.normal, share.normal)

res.ww <- df
```

```
ggplot(df.long) + geom_line(mapping=aes(x=date, y=Frac, color=Source))
```



We can get an estimate of the size of the market by dividing MAU by our share of the market, giving us the graph below. Based on this data, desktop market size has actually increased over this period, from about 2 billion to about 3 billion. These numbers do seem fairly high, but it's a straightforward calculation, and we just need to assume that there's no systematic bias in either of these measurements, then it would appear that the market is growing.

```
ggplot(df) + geom_line(mapping=aes(x=date, y=size))
```

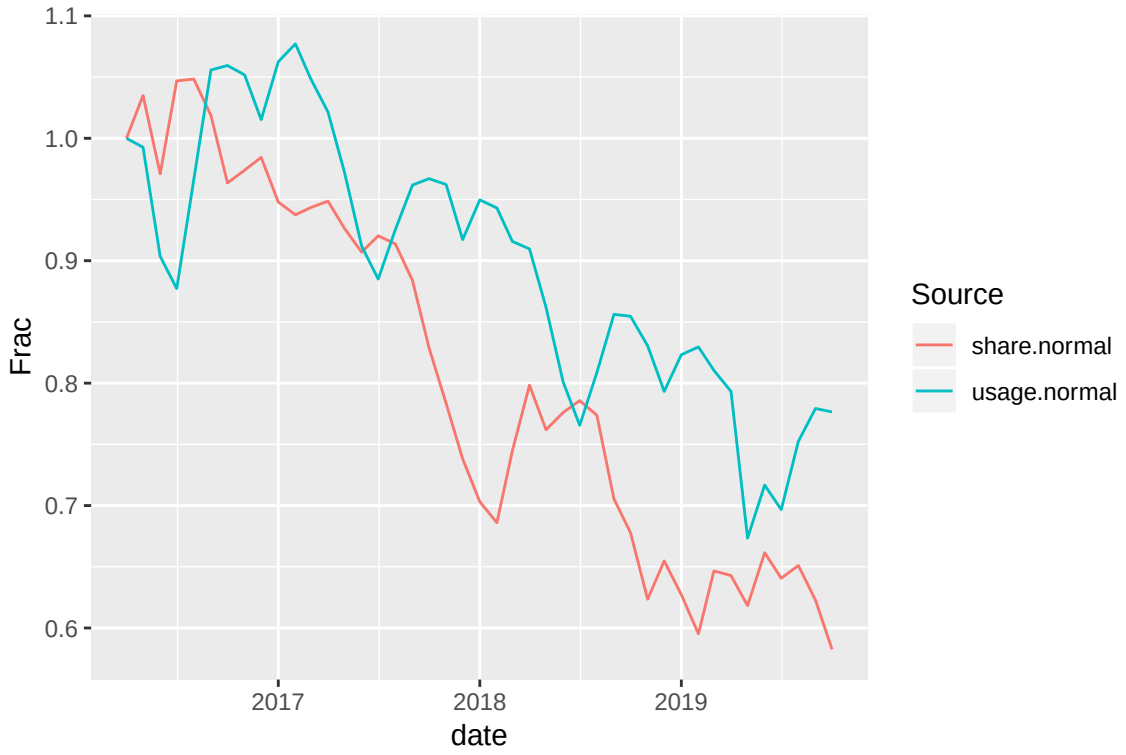


US Market Size

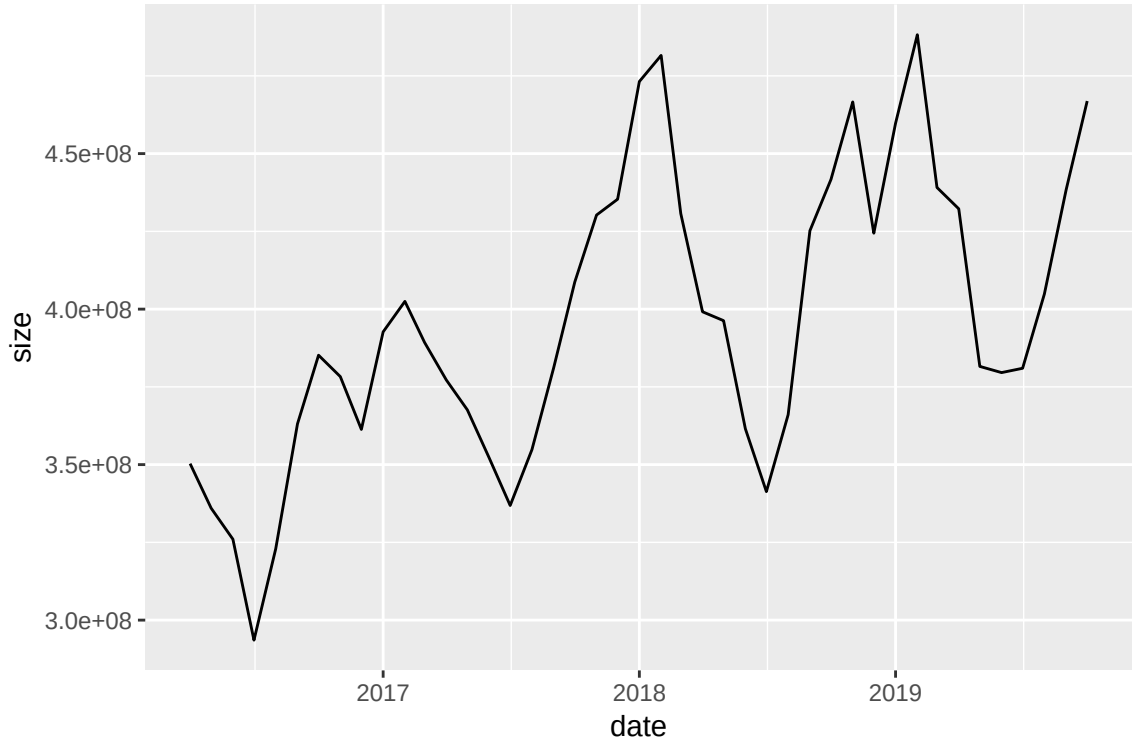
Here is the same treatment for the US, which shows the market size as approximately flat at ~400 million users.

```
df <- df.us

df$size <- df$usage/df$share
df$usage.normal <- df$usage/df$usage[1]
df$share.normal <- df$share/df$share[1]
df.long <- gather(df, key="Source", value="Frac", usage.normal, share.normal)

res.us <-df
```

```
ggplot(df.long) + geom_line(mapping=aes(x=date, y=Frac, color=Source))
```




```
ggplot(df) + geom_line(mapping=aes(x=date, y=size))
```



NetMarketShare

StatCounter is not the only source of market share data. This section runs the same analysis for worldwide data pulled from NetMarketShare. I was unable to get it to download a CSV, so I hand-transcribed it from their dashboard, which shows values if you hover over the points. This may have resulted in transcription errors and we should arrange to get machine readable data.

```
df <- df.nm

df$size <- df$usage/df$share
df$usage.normal <- df$usage/df$usage[1]
df$share.normal <- df$share/df$share[1]
df.long <- gather(df, key="Source", value="Frac", usage.normal, share.normal)

res.nm <- df
print(res.nm)
```

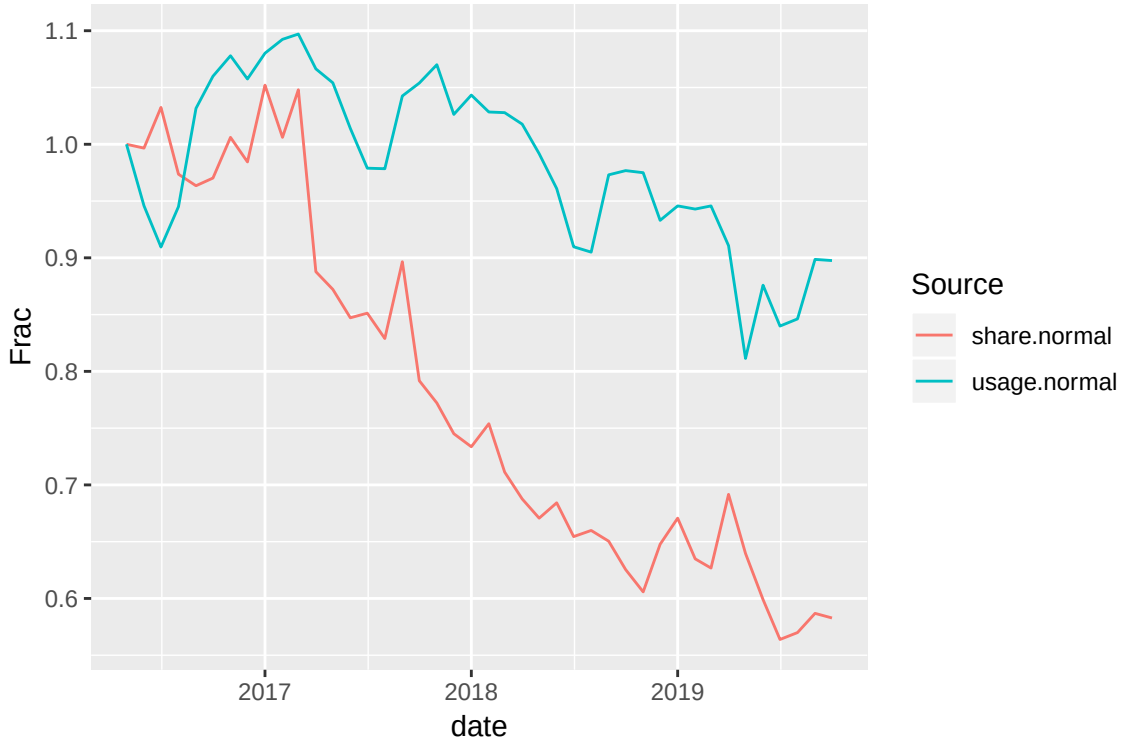
```
##           date     usage  share       size usage.normal share.normal
##  1: 2016-05-01 285634836 0.1479 1931270020    1.0000000    1.0000000
##  2: 2016-06-01 270154989 0.1474 1832801825    0.9458055    0.9966193
##  3: 2016-07-01 259790028 0.1527 1701309941    0.9095180    1.0324544
##  4: 2016-08-01 269944087 0.1440 1874611715    0.9450671    0.9736308
##  5: 2016-09-01 294668356 0.1425 2067848112    1.0316261    0.9634888
##  6: 2016-10-01 302749044 0.1435 2109749436    1.0599164    0.9702502
##  7: 2016-11-01 307906995 0.1488 2069267440    1.0779742    1.0060852
##  8: 2016-12-01 302053857 0.1456 2074545721    1.0574826    0.9844490
##  9: 2017-01-01 308542396 0.1556 1982920283    1.0801988    1.0520622
## 10: 2017-02-01 312032878 0.1488 2096995148    1.0924188    1.0060852
## 11: 2017-03-01 313358729 0.1550 2021669219    1.0970606    1.0480054
## 12: 2017-04-01 304595225 0.1313 2319841775    1.0663798    0.8877620
## 13: 2017-05-01 301098457 0.1290 2334096566    1.0541377    0.8722110
## 14: 2017-06-01 289622237 0.1253 2311430463    1.0139598    0.8471941
## 15: 2017-07-01 279620649 0.1259 2220974178    0.9789445    0.8512508
## 16: 2017-08-01 279485178 0.1226 2279650718    0.9784702    0.8289385
## 17: 2017-09-01 297783120 0.1326 2245724887    1.0425308    0.8965517
## 18: 2017-10-01 301033621 0.1171 2570739718    1.0539107    0.7917512
## 19: 2017-11-01 305643785 0.1142 2676390412    1.0700508    0.7721433
## 20: 2017-12-01 293145879 0.1102 2660125944    1.0262960    0.7450980
## 21: 2018-01-01 298014164 0.1085 2746674323    1.0433397    0.7336038
## 22: 2018-02-01 293748372 0.1115 2634514547    1.0284053    0.7538878
## 23: 2018-03-01 293589939 0.1052 2790778888    1.0278506    0.7112914
## 24: 2018-04-01 290690863 0.1017 2858317237    1.0177010    0.6876268
## 25: 2018-05-01 283261461 0.0992 2855458276    0.9916909    0.6707235
## 26: 2018-06-01 274484701 0.1012 2712299417    0.9609637    0.6842461
## 27: 2018-07-01 259847185 0.0968 2684371746    0.9097181    0.6544963
## 28: 2018-08-01 258498201 0.0976 2648547141    0.9049954    0.6599053
## 29: 2018-09-01 277948291 0.0962 2889275374    0.9730896    0.6504395
## 30: 2018-10-01 279010613 0.0925 3016330951    0.9768088    0.6254226
## 31: 2018-11-01 278482797 0.0896 3108066931    0.9749609    0.6058147
## 32: 2018-12-01 266494479 0.0958 2781779530    0.9329901    0.6477350
## 33: 2019-01-01 270130952 0.0992 2723094274    0.9457213    0.6707235
## 34: 2019-02-01 269337016 0.0939 2868338829    0.9429418    0.6348884
## 35: 2019-03-01 270126485 0.0927 2913985814    0.9457057    0.6267748
## 36: 2019-04-01 260124625 0.1023 2542762708    0.9106894    0.6916836
## 37: 2019-05-01 231764545 0.0946 2449942336    0.8114015    0.6396214
```

```
## 38: 2019-06-01 250185241 0.0886 2823761185    0.8758919    0.5990534
## 39: 2019-07-01 239904844 0.0834 2876556882    0.8399005    0.5638945
## 40: 2019-08-01 241701699 0.0843 2867161317    0.8461913    0.5699797
## 41: 2019-09-01 256677234 0.0868 2957110991    0.8986202    0.5868830
## 42: 2019-10-01 256378780 0.0862 2974231787    0.8975753    0.5828262
##           date     usage  share       size usage.normal share.normal
```

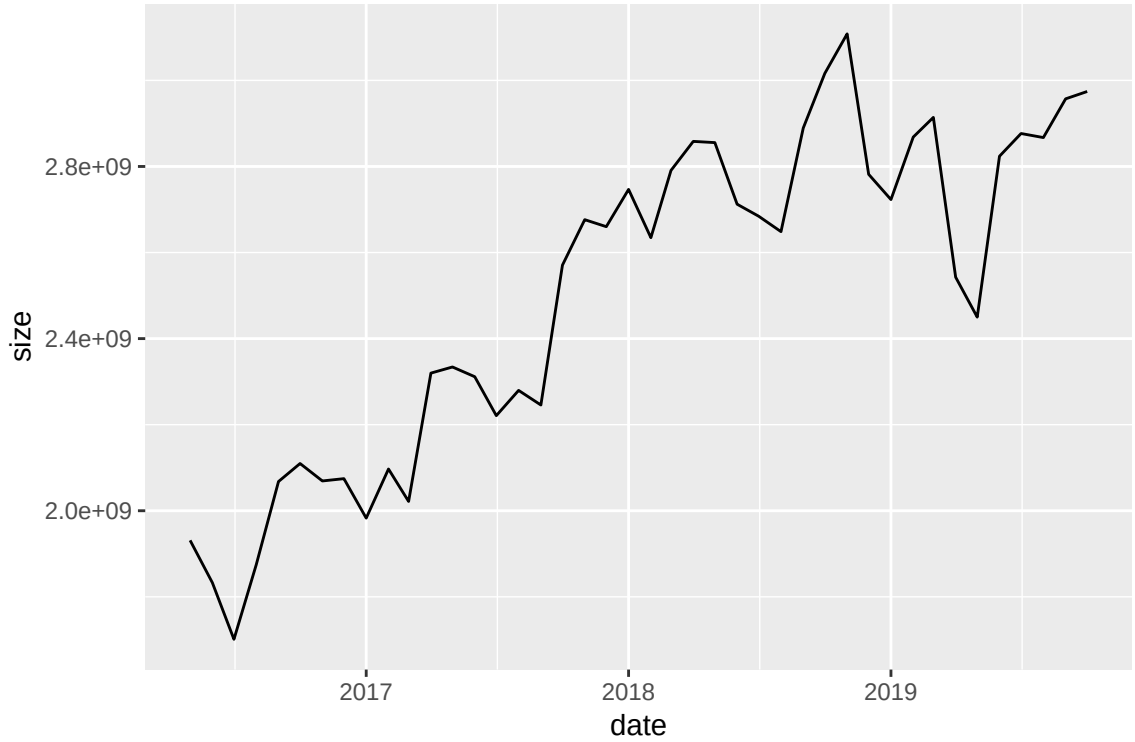
```
print(res.ww)
```

```
##           date     usage  share       size usage.normal share.normal
##  1: 2016-04-01 279628089 0.1562 1790192631    1.0000000    1.0000000
##  2: 2016-05-01 285634836 0.1596 1789691955    1.0214812    1.0217670
##  3: 2016-06-01 270154989 0.1555 1737331119    0.9661225    0.9955186
##  4: 2016-07-01 259790028 0.1543 1683668360    0.9290555    0.9878361
##  5: 2016-08-01 269944087 0.1531 1763188027    0.9653683    0.9801536
##  6: 2016-09-01 294668356 0.1500 1964455707    1.0537867    0.9603073
##  7: 2016-10-01 302749044 0.1459 2075044853    1.0826847    0.9340589
##  8: 2016-11-01 307906995 0.1489 2067877737    1.1011304    0.9532650
##  9: 2016-12-01 302053857 0.1495 2020427137    1.0801986    0.9571063
## 10: 2017-01-01 308542396 0.1485 2077726572    1.1034027    0.9507042
## 11: 2017-02-01 312032878 0.1481 2106906671    1.1158853    0.9481434
## 12: 2017-03-01 313358729 0.1497 2093244683    1.1206268    0.9583867
## 13: 2017-04-01 304595225 0.1453 2096319511    1.0892869    0.9302177
## 14: 2017-05-01 301098457 0.1417 2124900896    1.0767819    0.9071703
## 15: 2017-06-01 289622237 0.1398 2071689821    1.0357409    0.8950064
## 16: 2017-07-01 279620649 0.1382 2023304262    0.9999734    0.8847631
## 17: 2017-08-01 279485178 0.1373 2035580320    0.9994889    0.8790013
## 18: 2017-09-01 297783120 0.1360 2189581765    1.0649256    0.8706786
## 19: 2017-10-01 301033621 0.1304 2308540038    1.0765500    0.8348271
## 20: 2017-11-01 305643785 0.1255 2435408645    1.0930368    0.8034571
## 21: 2017-12-01 293145879 0.1221 2400867150    1.0483420    0.7816901
## 22: 2018-01-01 298014164 0.1187 2510650076    1.0657519    0.7599232
## 23: 2018-02-01 293748372 0.1154 2545479827    1.0504967    0.7387964
## 24: 2018-03-01 293589939 0.1160 2530947750    1.0499301    0.7426376
## 25: 2018-04-01 290690863 0.1178 2467664372    1.0395625    0.7541613
## 26: 2018-05-01 283261461 0.1155 2452480182    1.0129936    0.7394366
## 27: 2018-06-01 274484701 0.1144 2399341792    0.9816063    0.7323944
## 28: 2018-07-01 259847185 0.1123 2313866296    0.9292600    0.7189501
## 29: 2018-08-01 258498201 0.1096 2358560228    0.9244357    0.7016645
## 30: 2018-09-01 277948291 0.1094 2540660795    0.9939927    0.7003841
## 31: 2018-10-01 279010613 0.1014 2751583955    0.9977918    0.6491677
## 32: 2018-11-01 278482797 0.0910 3060250516    0.9959042    0.5825864
## 33: 2018-12-01 266494479 0.1005 2651686358    0.9530319    0.6434059
## 34: 2019-01-01 270130952 0.0950 2843483705    0.9660365    0.6081946
## 35: 2019-02-01 269337016 0.0872 3088727248    0.9631973    0.5582586
## 36: 2019-03-01 270126485 0.0958 2819691910    0.9660206    0.6133163
## 37: 2019-04-01 260124625 0.0978 2659760992    0.9302521    0.6261204
## 38: 2019-05-01 231764545 0.0995 2329291910    0.8288314    0.6370038
## 39: 2019-06-01 250185241 0.0976 2563373371    0.8947071    0.6248399
## 40: 2019-07-01 239904844 0.0952 2520008866    0.8579426    0.6094750
## 41: 2019-08-01 241701699 0.0952 2538883393    0.8643685    0.6094750
## 42: 2019-09-01 256677234 0.0954 2690537044    0.9179236    0.6107554
## 43: 2019-10-01 256378780 0.0925 2771662486    0.9168563    0.5921895
##           date     usage  share       size usage.normal share.normal
```

```
ggplot(df.long) + geom_line(mapping=aes(x=date, y=Frac, color=Source))
```




```
ggplot(df) + geom_line(mapping=aes(x=date, y=size))
```

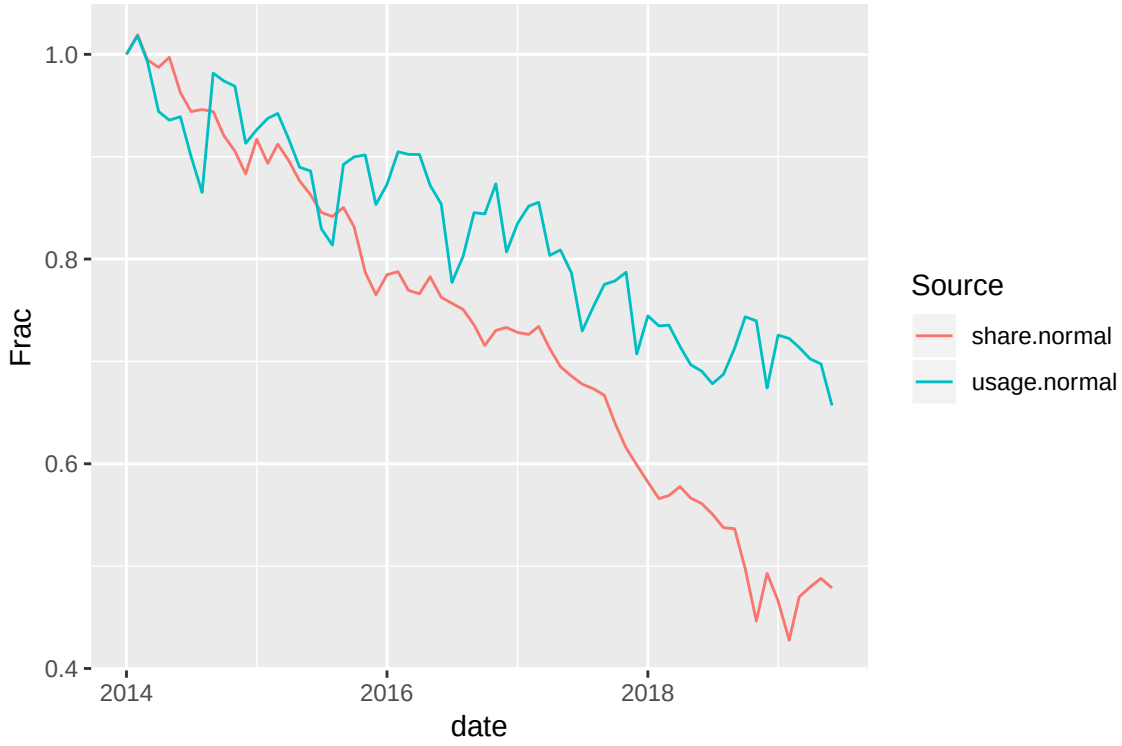


ADI

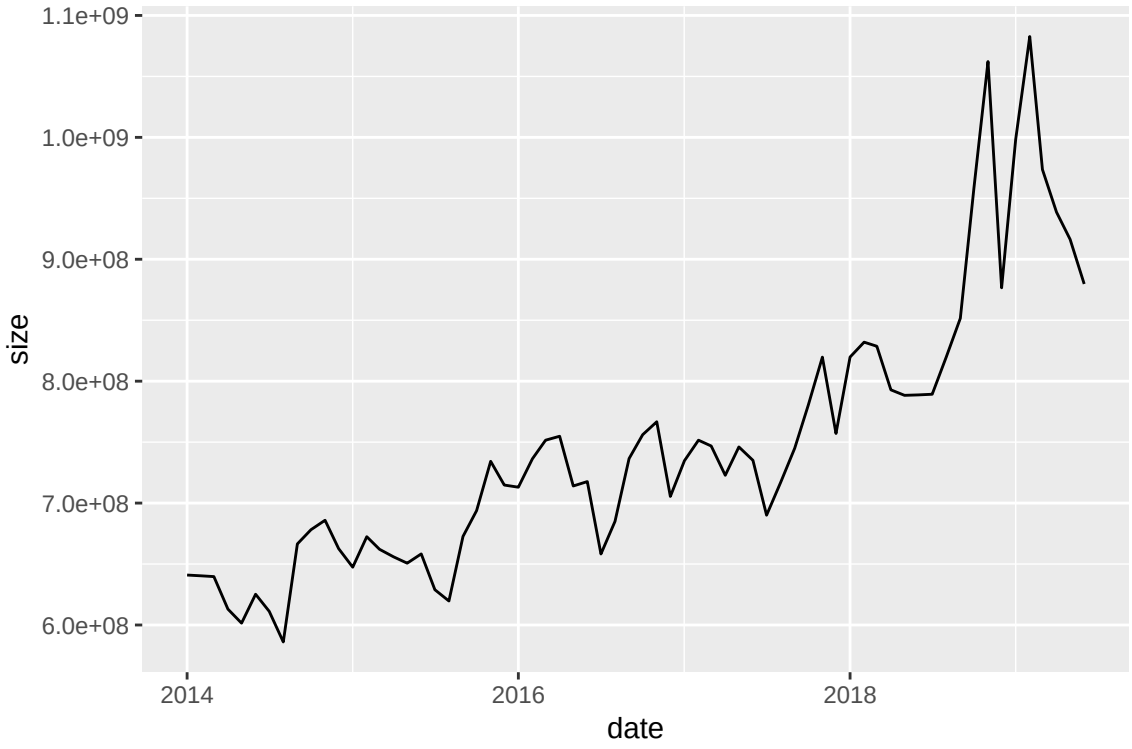
And finally, here's the same treatment for Average Daily Installs (ADI), which is based on the blocklist ping rather than on telemetry data.

```
df <- df.adi

df$size <- df$usage/df$share
df$usage.normal <- df$usage/df$usage[1]
df$share.normal <- df$share/df$share[1]
df.long <- gather(df, key="Source", value="Frac", usage.normal, share.normal)

res.adi <- df
```

```
ggplot(df.long) + geom_line(mapping=aes(x=date, y=Frac, color=Source))
```




```
ggplot(df) + geom_line(mapping=aes(x=date, y=size))
```



Summary

Finally, we can compare the each of these series by normalizing to the value at the beginning of the MAU series (which starts later than ADI). Because ADI is an unsmoothed value, we take the arithmetic mean of ADI for each month. As before, we attribute it to the first day of the month.

TECHNICAL NOTE: The netmarket share data is normalized against its first appearance, in June 2016, one month later than the MAU series.

```
res.ww$size.normal <- res.ww$size/res.ww$size[1]
res.us$size.normal <- res.us$size/res.us$size[1]
res.adi$size.normal <- res.adi$size/(res.adi[date == min(res.ww$date)]$size[1])
res.nm$size.normal <- res.nm$size/res.nm$size[1]
res.ww$series <- "WW MAU (StatCounter)"
res.us$series <- "US MAU (StatCounter)"
res.adi$series <- "ADI (StatCounter)"
res.nm$series <- "WW MAU (NetMarketShare)"
print(names(res.nm))
```

```
## [1] "date"         "usage"        "share"        "size"
## [5] "usage.normal" "share.normal" "size.normal"  "series"
```

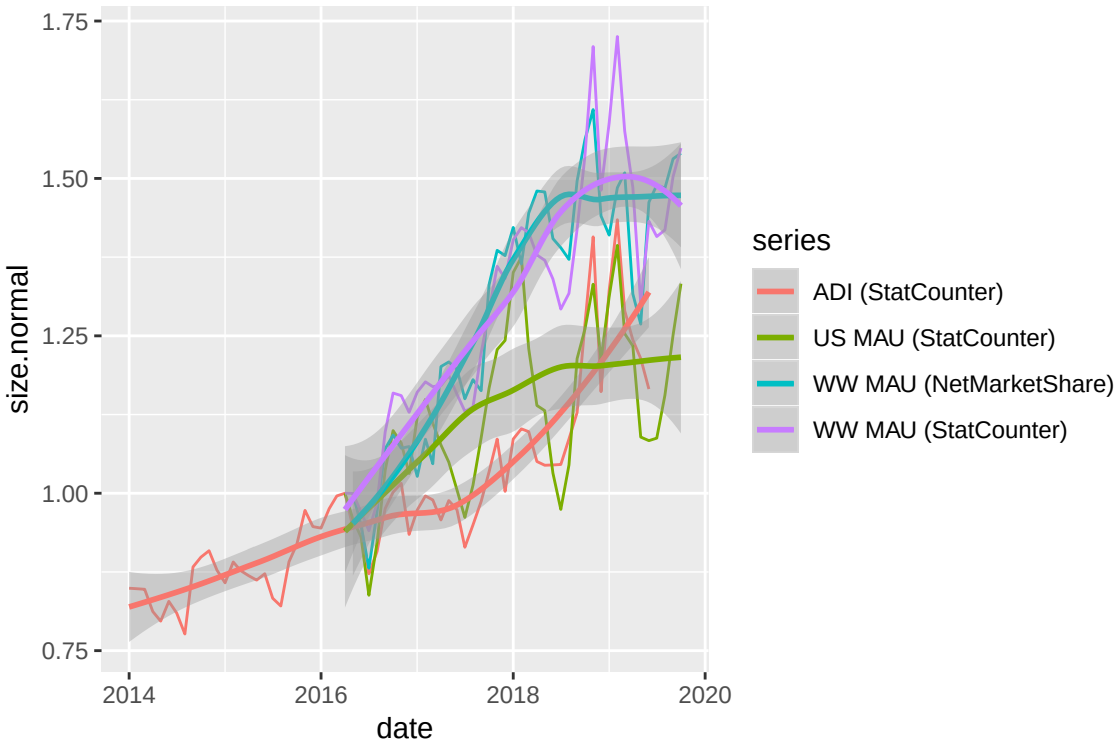
```
print(names(res.ww))
```

```
## [1] "date"         "usage"        "share"        "size"
## [5] "usage.normal" "share.normal" "size.normal"  "series"
```

```
res <- rbind(res.ww, res.us, res.nm, res.adi)
```

```
ggplot(res) + geom_line(mapping=aes(x=date,y=size.normal, color=series)) + geom_smooth(mapping=aes(x=da
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



It's probably a mistake to read too much into these data sets (and in particular the projections implied by smoothing are not reliable due to seasonality), but a few points are worth noting. All of this data generally show a market increase over the past few years (causeed by market share decline outpacing absolute decline), coupled with big spikes adound the beginning of 2019. These reflect two big negative spikes in market share around the same time, so may be measurement error in the market share estimates, and in any case will recur for any absolute usage series we run against these market share numbers. The NetMarketShare data does not show these big spikes but overall tells a very similar picture to that for the StatCounter WW data.

In general, this is a somewhat surprising result as we had expected the desktop market to be generally flat to in decline, and so more investigation is definitely required.

Sanity Check

There are a lot of potential sources of noise in this data. As a sanity check, let's compare the US data to an independent source, which is the installed base of desktop and laptops in the US from Statista. [https://www.statista.com/statistics/670172/united-states-installed-base-desktops-laptops-tablets/]. This shows the market as being nearly flat at around 310 million for the past few years, projecting to be 305 million in 2022, so at least we are on the same order of magnitude.

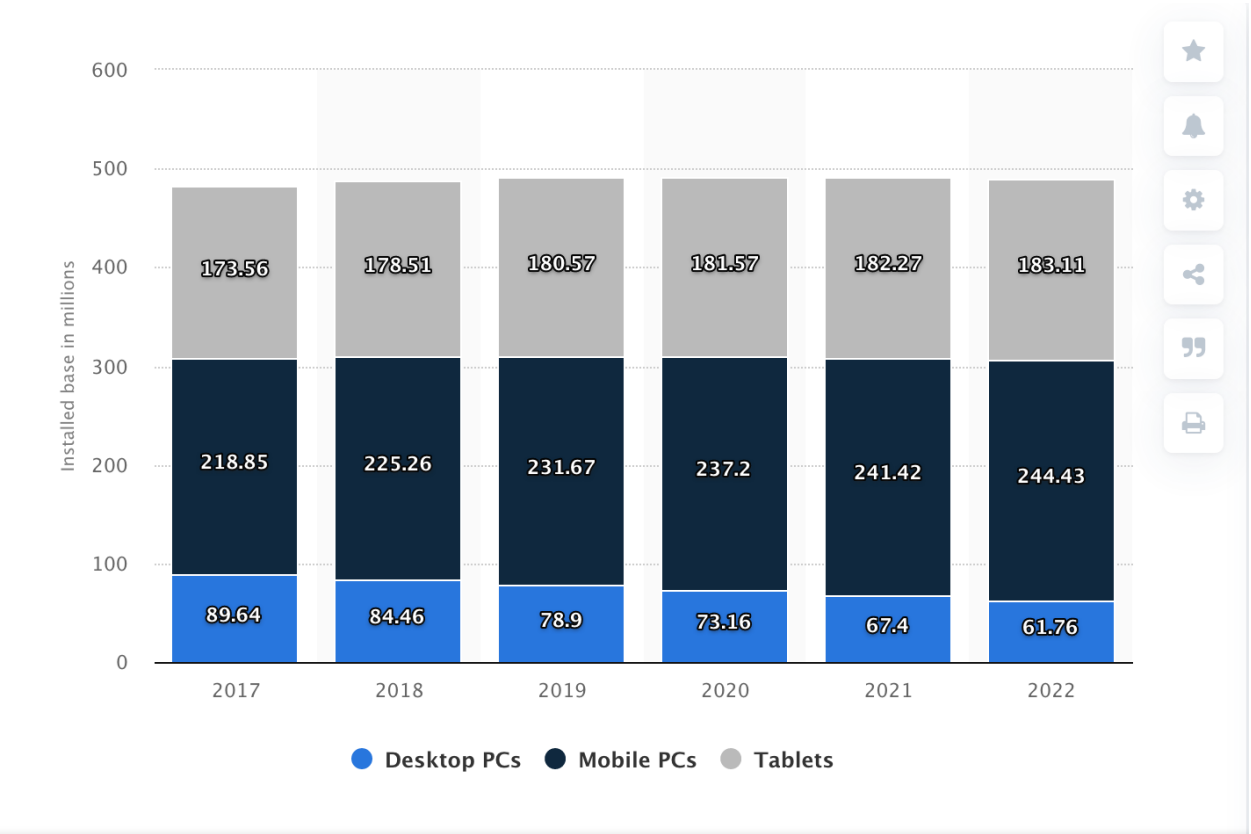


Figure 1: US Desktop and Laptop Market Size (Statista)