What makes an R-package popular?

Yiwen Zhang

2021-06-11

Contents

Preface

As an R user, have you ever thought about what kind of information is hidden behind the number of downloads for R and R-packages? Why would there an abnormal download value? When choosing which package to use, we often probably tend to use popular ones. But what kind of packages are popular? And What factors will linked to the popularity of an R-package? I suppose these are questions that most R users will be curious about. Therefore, based on the existing findings, this report answers the above questions to a certain extent by taking an exploratory data analysis.

This is written as part of the ETC5543 research project supervised by Drs Emi Tanaka and Hien Nguyen. Due to the limitations of my programming ability, I encountered some obstacles during this research. But this is also a very precious opportunity for me to cultivate my research thinking and improve my confidence in coding.

I would like to express my very great appreciation to Drs Emi Tanaka and Hien Nguyen for their valuable and constructive suggestions during the planning and development of this research work. As my supervisors, their patient guidance, enthusiastic encouragement and willingness to give their time so generously have been very much appreciated. And my special thanks are also extended to Dr Emi Tanaka for her generous help on writing. I would like to also thank Professor Rob J Hyndman, for holding every two weeks' meeting to give us recommendations.

I hope you enjoy your reading.

Chapter 1

Abstract

This research seeks to analyze the popularity of R-packages and figure out the influential factors. Data are closely related to social and economic activities in our daily life, so data mining and analysis will definitely benefit us. In many statistical and data analysis tools, this research focuses on R in this project, and aim to explore the popularity of R-packages from CRAN. To achieve this, it is assumed in this research that the download volume is a relatively reliable and simple measurement for the popularity of R-packages, and two main parts of analysis has been carried out, one is to uncover the information behind the download count logs, the other is to explore the factors that will linked to the download volume. The techniques applied in this research is mainly R, together with a small part of Python on data scraping. By obtaining the daily and recent half a year's total download count of 17,699 R-packages and R itself, this research analyzes their pattern characteristics, as well as the relationship between the release date, update times, number of commits on Github, name length and alphabetic order of the name. Finally, this research also gives the changes of R-user preferences on 1st April from 2013 to 2021, to some extent.

Moreover, results show that the daily download of R-packages has a strong weekly seasonality and unusual spikes caused by repeat download, update or server issue. The most stably popular R-packages during 2017 to 2019 is about JAVA dependency. And an earlier release date is more likely to bring a higher download volume for R-packages. That is similar to update times and commits number on Github repository. However, the alphabetical order of R-package name contributes little to the download counts. And over half of the R-packages on CRAN tend to have shorter names.

KEY WORDS : R-package popularity, CRAN download count, Web scraping, Github API, EDA

Chapter 2

Introduction

Data, which has penetrated into every industry and business field, has become an increasingly important production factor and consumption in our lives. However, data itself must be processed to uncover the information within it. For this reason many statistical tools were born, such as SPSS, Python, R and so on. In this project, we focus on the R language (R Core Team 2021).

R is a language and environment for statistical computing and graphics which can be extended easily via packages and provide an Open Source route to participation in statistical methodology. It is available as Free Software under the term of the Free Software Foundation's GNU General Public License (*The GNU Operating System and the Free Software Movement* 1991). R is one of the most popular statistical languages and has been ranking competitively even among the general purpose programming languages (peaking 8th in August 2020 in *Latest news* n.d.). The TIOBE Programming Community index aggregates several search engines to derive a metric of the popularity of a programming language. It is important to note, however, that programming languages ranked higher than R are mostly general purpose languages and therefore naturally has a larger user base whereas R is exclusively for data analysis and statistical computation alone.

Today, R is greatly enhanced by over 17,699 R-packages contributed by hundreds of developers all over the world. However, when R originally appeared in August of 1993 with its first official release in June of 1995 (Ihaka 1998), the contributions were managed by only a small group of core developers. In April of 1997, Comprehensive R Archive Network (CRAN) was established as the official R-packages repository with 3 mirror sites. Now the source repositories to install R-packages have expanded to Bioconductor, Gitlab, GitHub, R-Forge and 106 CRAN mirrors in 49 regions. Of all the

CRAN mirrors, the daily download count for each package is only readily available from the RStudio CRAN mirror. In this report, we will be focusing only on one CRAN mirror via RStudio, albeit the most popular one, for exploratory data analysis. Henceforth, all download counts of packages refer to the download from this RStudio CRAN mirror.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R (n.d.).

The most intuitive embodiment of popularity is the total number of downloads, although it does not directly reflect the number of users, because it includes the repeated downloads by the same user, updates and test downloads caused by the server. Briefly, some works that use these download counts include: the R-package `adjustedcranlogs` (Morgan-Wall 2017) attempts to correct this download inflation using a heuristic approach; package `packageRank` (Peter Li 2021) provides a way to compute the rank percentile for packages and filter the invalid downloads including small and medium size logs. What's more, there are also some packages such as `pkgsearch`(Csárdi and Salmon 2020) and `Visualize.CRAN.Downloads`(Ponce 2021) providing some visualization methods to explore the package download trend or for convenient searching. More details are explained in Section 5.

Based on that, researchers have started to study the popularity of R-packages from CRAN. And we still assume in this project that, the download amount is a relatively reliable and simple measure of packages' popularity. Back to the previous studies, the R-package `adjustedcranlogs`(Morgan-Wall 2017) finds that there are spikes in downloads due to automatic re-downloads and package update, and it also provides a method to remove these download logs. Some of them are function extensions based on previous packages, and some of them propose their own new functions, but generally speaking, they are inclined to be more tool like to help users explore the characteristics of package download logs from as many dimensions as possible.

As for us, the purpose of this project is to explore the download logs of R-packages on CRAN and analyze the relationship between some influential factors with it to help users and developers better understand the download amount pattern, and also figure out what would determine a package's popularity to some extent.

- CRAN mirror is a website containing differently located servers, which aims to facilitate people from different regions and countries to access CRAN more smoothly and quickly. And each server is called a mirror (*CRAN Mirrors* n.d.).

- CRAN task view : It aims to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic (n.d.).

2.1 Other work

`adjustedcranlogs`: More specifically, any unusual download spikes are smoothed over and the upward download bias is removed stochastically by subtracting the minimum download count from a random sampling of packages; if this corrected download results in a negative value then this is adjusted to be zero.

Chapter 3

Data

The main source of data used in this report is the download logs from the RStudio CRAN mirror site https://cran.rstudio.com/. These log files are created for every instance of download of an R-package via the RStudio CRAN mirror, then these log files are processed daily into CSV files that contain the following variables with the name of header in brackets:

- Date (`date`),
- Time in UTC time zone (`time`),
- Size of the file in bytes (`size`),
- Version of R used to download the package (`r_version`),
- Architecture type for R (i386 = 32 bit, x86_64 = 64 bit) (`r_arch`),
- Operating System (darwin9.8.0 = mac, mingw32 = windows) (`r_os`),
- Package (`package`),
- Country in two letter ISO country code (`country`), and
- Anonymised daily unique id (`ip_id`).

A similar log file is also created for every download of R from the RStudio CRAN mirror with the processed log file generating a CSV file that contains the same variables except `r_arch` and `package`, and `r_version` and `r_os` are named as `version` and `os`. These CSV files are hosted at http://cran-logs.rstudio.com/ and updated daily with data available from 1st October 2012.

The log files of a particular day is processed and compressed into a single CSV file of about 40 megabytes (file sizes of earlier years are much smaller due to lower number of download logs). As there are over 700,000 CSV files, a simple estimate of the size of the data is 28 terabytes - far exceeding typical portable hard drives which are 1-4 terabytes.

The summarised version of data, where the data shows the total daily down-

load count for each package, is accessible using the `cranlogs` R-package. The `cranlogs` package accesses this summary data through the web application programming interface (API) maintained by r-hub (R-Hub n.d.[a]).

Chapter 4

Methods

In this project, we extracted R-package names from CRAN and got summary daily download logs of R-packages through web Application Programming Interface (API) maintained by r-hub(R-Hub n.d.[a]). And we also collected release dates of R-packages through CRAN. In addition, we also tried to scrap the number of commits in Github repository for R-packages. Then we constructed eight exploratory data analysis sections:

- we explored the daily download of all the packages on CRAN and figured out the number of R-packages occupied by different downloaded groups;
- we analyzed the daily download trend of R itself and the found out the most popular version of R;
- we compared the top 15 downloaded R-packages on CRAN on 1st April. from year 2013 to 2021 to see how the user preference changes;
- we figured out the relationship between initial release date and the total download count for last half a year;
- we compared two closely related packages `fable` and `forecast` to see the difference between their MA (moving average);
- we studied how the number of commits of master (main) branch on Github repository may influence the total download count for last half a year;
- we explored the relationship between the number of updates of R-packages and its download count;
- we analyzed the name length pattern for both CRAN task view packages and all the R-packages on CRAN;

- we explored how the alphabetical order of R-packages would related to the download count and its statistic features (variance, median).

Chapter 5

Results

5.1 Daily download of R-packages

Finding 1: There was unusual download activity in one day of 2014 and 2018.

In this first section, we studied the daily downloads of CRAN R-packages from 2012-10-01 to 2021-06-07. The data was obtained from the `cranlogs` package(Csárdi 2019), which includes a summary of the download logs from the RStudio CRAN mirror. The daily download data for CRAN R-packages are available from 1st October 2012. Examination of this data showed two unusual observations in 2014 and 2018 as shown in Figure 5.1. The one happening in 2014 was on 2014-11-17, which was Monday, while the other one happening in 2018 was on 2018-10-21, which was on Sunday.

Then let's have a closer look into these two spikes. First, we focused on the spike on 2014-11-17. From Table 5.1, we could see that the downloads of top downloaded R-packages on this day differs little, so it's not due to certain package.

Table 5.2 shows the downloads from different countries. It is obvious that downloads from Indonesia is much more than any others, which indicates the most downloads are from Indonesia.

Furthermore, we also checked the IP address in Table 5.3, downloads from `ip3758` is much higher than others. So, it seems that most of the downloads are owing to one certain IP.

Next, let's turn to the one in 2018. Table 5.4 shows the downloads from

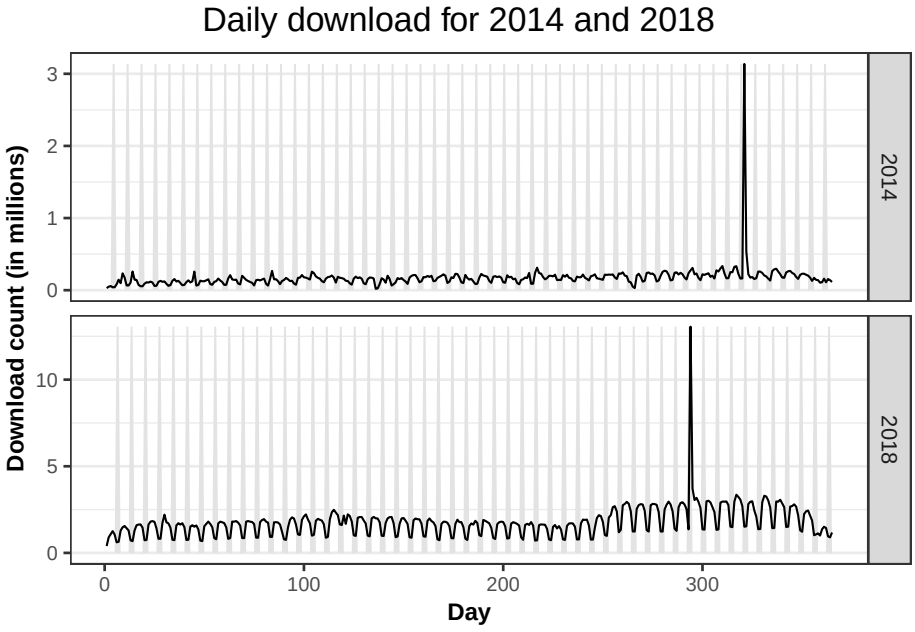


Figure 5.1: Unusual download spikes on 2014 and 2018.

Table 5.1: The total downloads of each R-package on 2014-11-17

package	n
BayHaz	767035
clhs	660298
GPseq	394840
OPI	382518
YaleToolkit	370513
survsim	224994
BAT	40592
Rcpp	3509
ggplot2	3167
plyr	3150

Table 5.2: The countries downloading from CRAN on 2014-11-17

country	n
ID	2863576
US	96336
CN	32729
DE	14548
FR	11860
GB	10491
IN	8635
HK	8090
BE	7720
KR	6794

Table 5.3: The IP address downloading from CRAN on 2014-11-17

ip_id	n
3758	2863432
11536	6244
11725	5992
16385	5991
534	5986
3784	5983
18519	4511
80	2124
27	1892
464	1375

Table 5.4: The total downloads of each R-package on 2014-11-17

package	n
tidyverse	11692582
Rcpp	16263
stringi	13981
rlang	13796
ggplot2	13306
dplyr	13081
glue	12593
digest	12302
stringr	11505
fansi	11275

Table 5.5: The countries downloading from CRAN on 2014-11-17

country	n
US	12140853
NA	179847
GB	76624
IN	51502
CN	46095
TR	36590
AU	35078
DE	32837
CA	31125
KR	30469

tidyverse is much higher than others with nearly three orders of magnitude.

As for the country, from Table 5.5 we could know that US is much higher than any other country.

Finally, the most interesting finding is in IP address displayed in Table 5.6. Several consecutive IPs have highly distinguished downloads. It seems that they are from same person, or it is also probably a server test issue in the same short period of time.

To sum up, we found that these two unusual spikes have one thing in common, that is, most of the downloads came from a specific country. The difference is that in 2014, a large number of downloads came from several different R-packages, while in 2018, they came from only one package `tidyverse`. In addition, in 2014, a large number of downloads came from

Table 5.6: The IP address downloading from CRAN on 2014-11-17

ip_id	n
266	3034720
263	2457383
655	2099321
264	1557640
267	1406876
265	1032535
2	179711
268	99932
112	34397
3296	17223

one IP, while in 2018, they came from several consecutive IPs, At this point, we guess it should come from the same person, and it is likely to be sever test issue, for it may be not necessary or reasonable for an individual to generate such a large quantities of downloads in one day.

Finding 2: There is an increasing number of downloads over time. This likely attests to the growing number of R users.

Figure 5.2 shows the download trend of all R-packages on CRAN over time after fixing the unusual spikes. It shows an upward trend over time, and the variance also increases with the download count, which means the volatility of the data is increasing.

Finding 3: Weekends have a lower download than weekdays.

To have a closer look at the weekly pattern, figure 5.3 shows the daily downloads of all CRAN R-packages from the RStudio mirror with the grey areas highlighting the weekend.

To be more specific, we could know that except for 2012 and 2013, the patterns of other years are very similar, that is, they all show strong weekly seasonality. To be more detailed, in 2012, the download logs showed an overall upward trend, because more and more users began to download R-packages from CRAN after its open. In the following years, there is no obvious trend in download volume, but a strong seasonality, which indicates that in a week, the total downloads always increases first then decreases, and reaches the lowest at the weekend. Although the pattern of 2013 is more volatile, it still conforms to that. We think for 2013, that is because

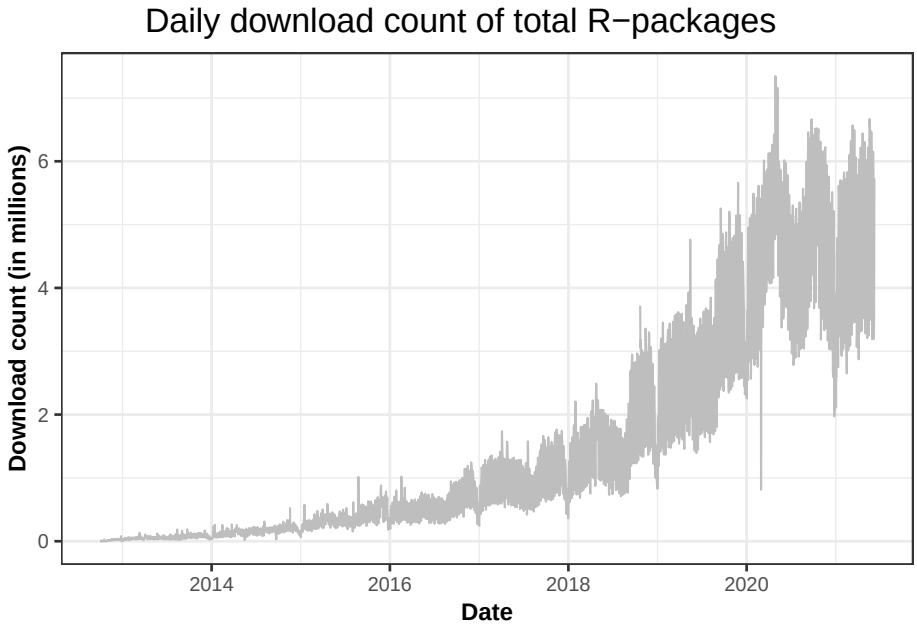


Figure 5.2: The download trend of all R-packages on CRAN over time.

CRAN is only open for a short time at this time, and the amount of data downloaded is not adequate to show its download pattern very clearly. Considering this, we could see that after 2016, the pattern of each year is quite consistent, for the total download has been increasing year by year. Back to weekly seasonality, that is because people are more likely to download and use packages in weekdays, and rest on weekends. And that's why the trough of download curve always occurs on weekends. In addition, we could also notice that the lowest downloads across the year are always at the end of December and the beginning of January, probably due to the Christmas and New Year's holidays. What's more, the downloads is on the rise from August to October and from February to April, which covers the start of semester for most universities.

As there are many fluctuation in daily download pattern which is due to calendar effect and server issue of CRAN mirror, we then applied a model called STL decomposition explained in *Forecasting: Principles and Practice (3rd ed)* (n.d.), to smooth the curve for all the R-packages.

And this can be applied to any R-package to adjust the daily download pattern. In this case, we selected two packages `fable` and `forecast` as an example in Figure 5.5. It can be seen that the pattern is smoother after removing the seasonality and ignorance of extremum possibly caused by

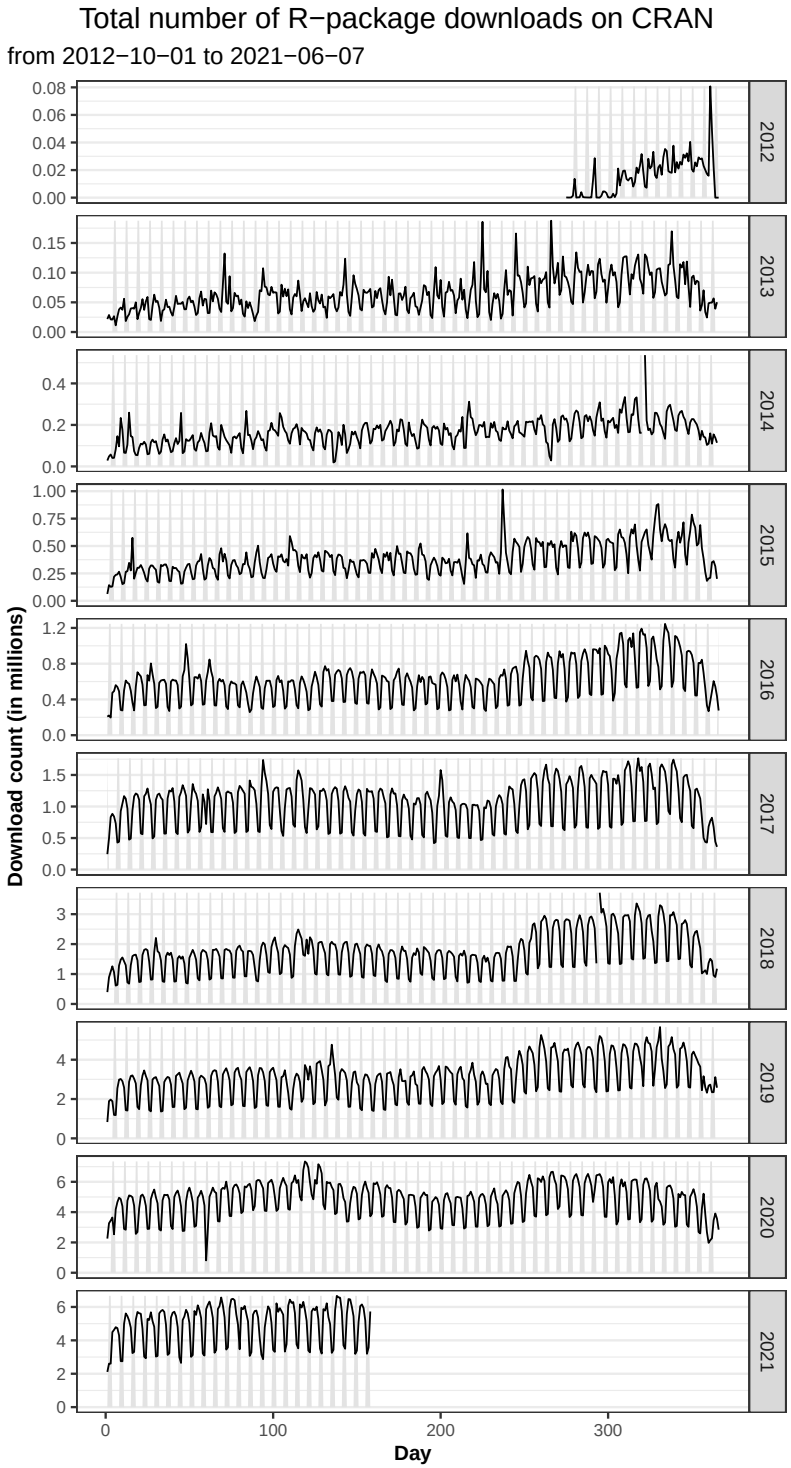


Figure 5.3: The figure shows the total downloads of all R-packages on CRAN would decrease on weekends.

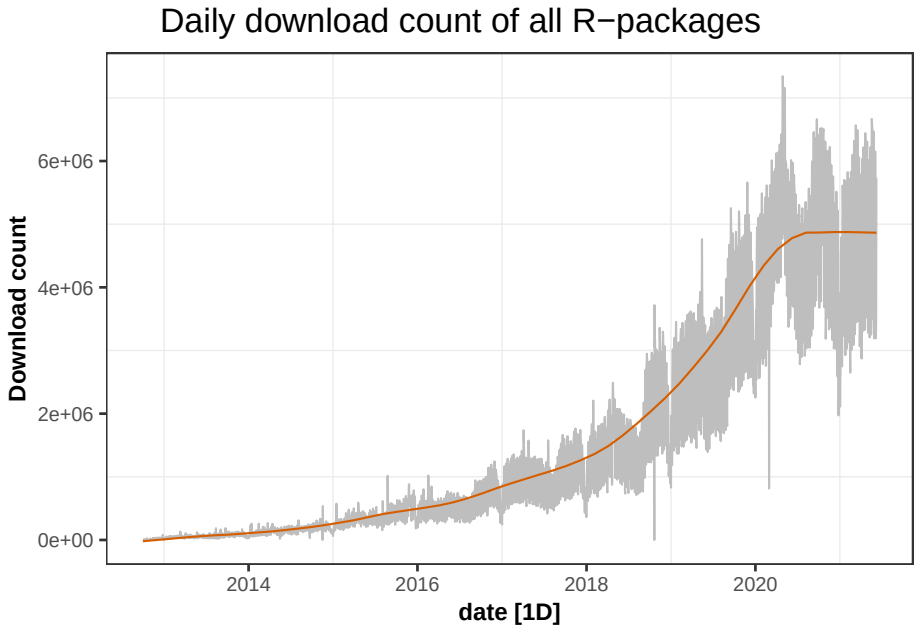


Figure 5.4: The figure shows the total downloads of all R-packages on CRAN after smoothing.

repeated downloads, updates and test downloads from the server.

Figure **??** shows the distribution and the median of the downloads between weekday and weekends. The distribution of weekdays and weekends are quite different. Weekends are wider and shorter, while weekdays are thinner and higher, because the total download of data on weekends is less than that on weekdays. And in 2012, the median and interquartile range of download logs are not very distinguished between weekdays and weekends, for the data volume was not adequate at this time as mentioned above. But after 2013, the gap between the two becomes more and more obvious, that is, the median downloads of working days is significantly higher than that of weekends, and the overall number of data is also significantly higher than that of weekends as well. But interestingly, the lower adjacent sometimes occurs on weekends, such as in year 2014, 2015, 2018, 2019 and 2021, while sometimes in weekdays, such as in year 2012, 2013, 2016, 2017 and 2020.

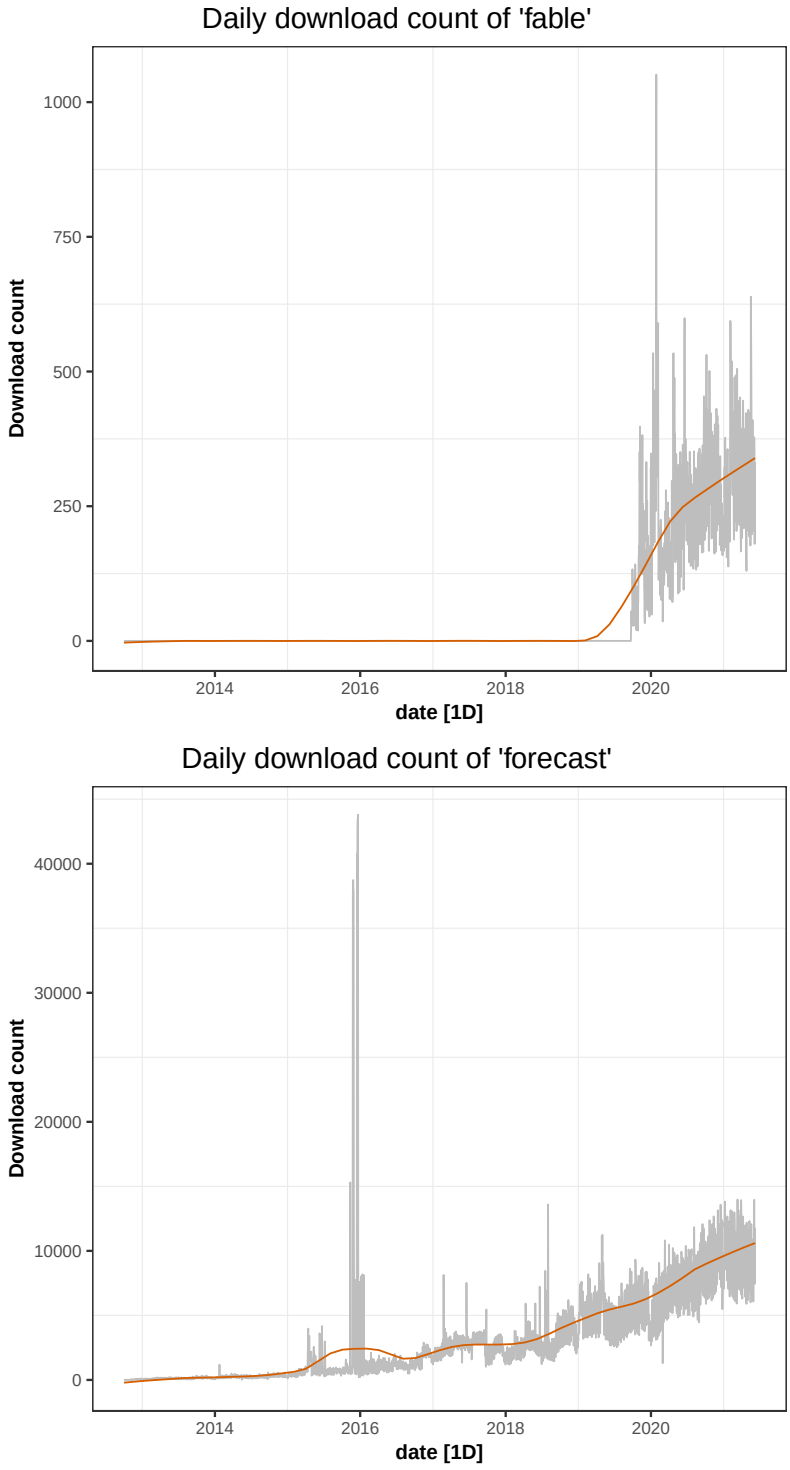
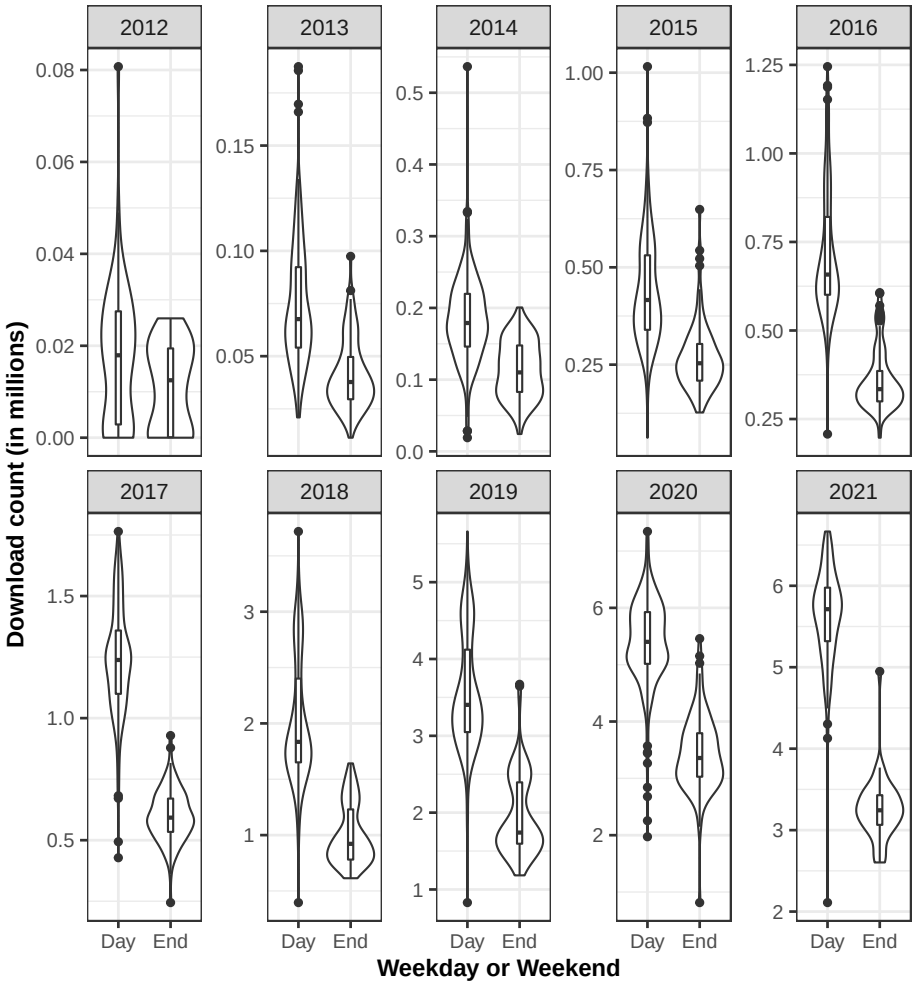


Figure 5.5: The figure shows the daily downloads of `fable` and `forecast` on CRAN after smoothing.



Finding 4: Top 10% downloaded R-packages share nearly 90% cumulative download count of the whole.

From the previous analysis, we could see that the cumulative download count of R-packages shows an increasing trend. It would be perfect equality if every R-package had the same download count – the last 20% downloaded R-packages would gain 20% of the total download count or the top 60% downloaded R-packages would get 60% of the total download count. But we know from experience that this is obviously impossible, so here we introduced Lorenz curve(Pettinger n.d.) to show the respective number of R-packages of different download levels (groups defined by quantiles of download count). In this way, we could figure out how many download counts contributed by different downloaded R-packages.

Figure 5.2 shows cumulative download count against each downloaded group. It can be seen that most of the download counts come from the top 10% downloaded R-packages. At the same time, we could also observe that the Gini value is close to 1, which indicates that the download volume among groups is very unbalanced. In fact, the download volume of the top 10% group is extremely distinguished from that of the following groups. It's not hard to understand that this group should contain some R-packages with high popularity and large quantities of users. For example, if we extracted the first 10 packages of this group in Table 5.1, we could find that they are all quite famous and frequently-used ones.

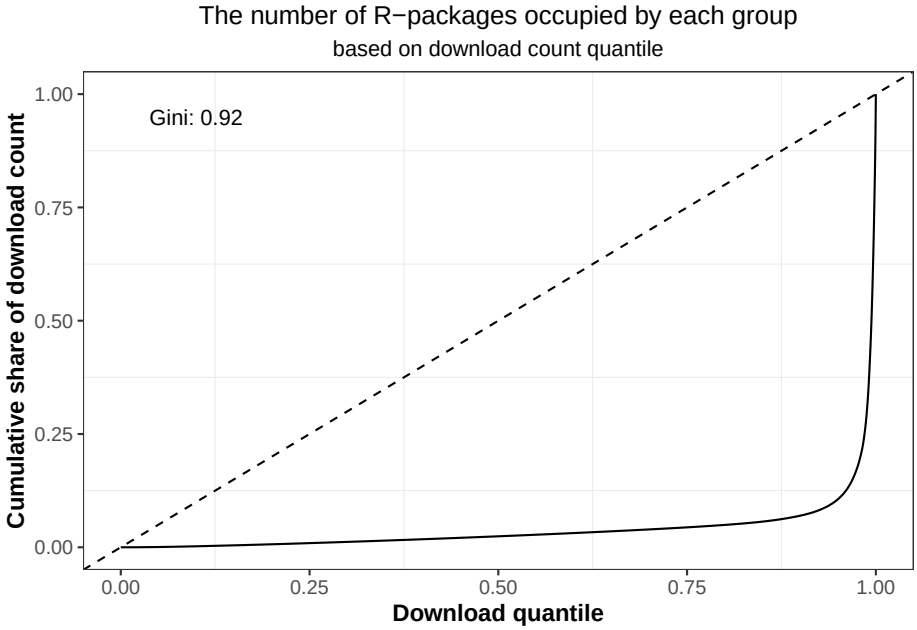


Figure 5.6: Percentiles of the download count against cumulative download count of R-packages at or below that percentile.

\begin{table}

\caption{First 10 R-packages of top 10% downloaded group}

package	total
rlang	15572507
vctrs	13544857
dplyr	12739206
ggplot2	12670952
jsonlite	12627542
lifecycle	11124212
tibble	10935860
magrittr	10312021
pillar	9566463
glue	9534999

\end{table}

5.2 Daily downloads of R

In this section, we studied the daily downloads of R language itself from 2012-10-01 to 2021-06-07. The data was obtained from the `cranlogs` package (Csárdi 2019) as well.

Finding 1: The number of downloads shows an upward trend over time, and the variance also increases with the downloads.

Figure 5.7 shows the download trend of R on CRAN over time. It shows an upward trend over time, and the variance also increases with the download count, which indicates the volatility of the data is increasing as well. And that is resulted from the pump of R users. As for the significant spikes, it is probably quite similar to the situation of R-packages, which is due to repeated downloads, weekly calendar effect and server test.

Finding 2: The most used operation system for R users is Windows OS.

Figure 5.8 shows the comparison of operation system for R users. The number of Windows users ranks first, followed by MacOS and SRC. For Microsoft has traditionally dominated the desktop and laptop market. And Microsoft officially claims there are 400 million active users of Windows 10 itself, while Apple revealed that there are now nearly 100 million active Mac users.(Warren 2017)

Finding 3: The most popular version of R is 3.2.1

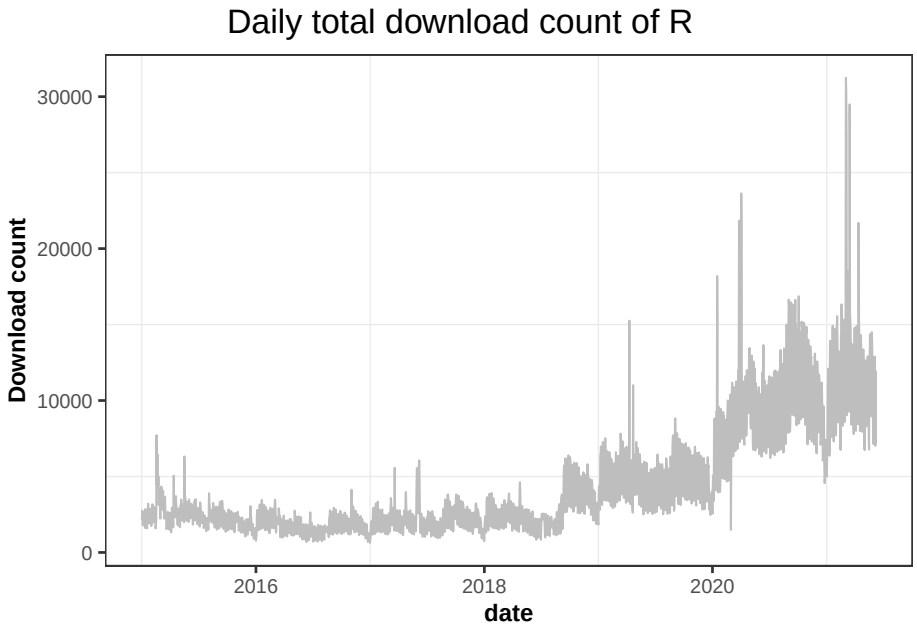


Figure 5.7: The download trend of R on CRAN over time.

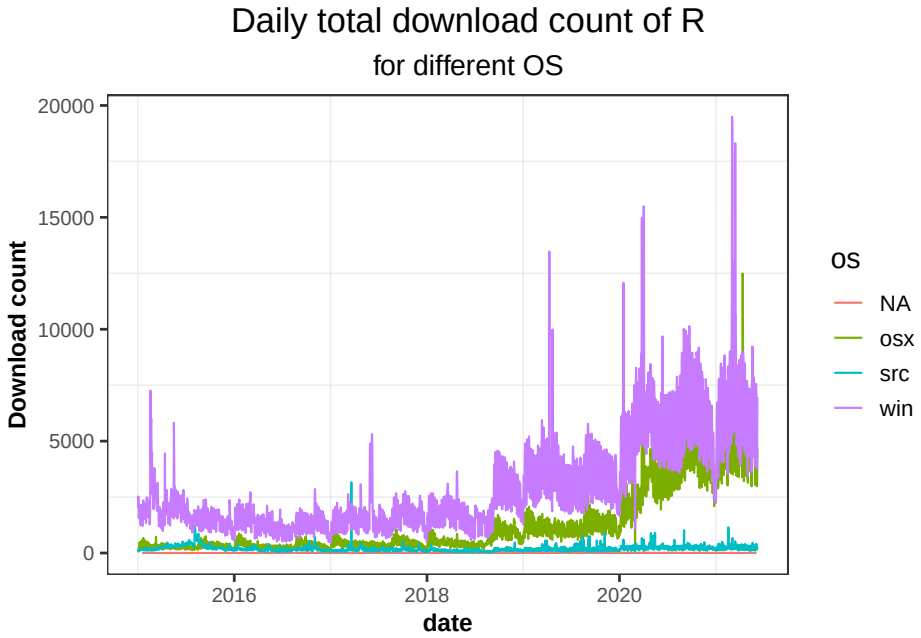


Figure 5.8: The download trend of R on CRAN over time.

And Table 5.7 shows the number of downloads of different versions of R. The most downloaded one is 3.2.1 with 4526 times, followed by 3.3.3 with 4305 times and 3.1.2 with 3958 times. So, most users tend to install the latest version of R to get the newest update information.

In conclusion, the daily download trend of R is quite similar to that of R-packages, and they are both significantly affected by weekly calendar issue, unusual downloads from CRAN mirrors. But in general, they both tend to increase with the growing of R users.

5.3 Daily top 15 downloaded R-packages

In this section, we studied the top 15 downloaded R-packages from 2013-04-01 to 2021-04-01 to see how user preferences has been changing. The data was obtained from CRAN website [http://cran.rstudio.com/web/packages/packages.rds].

We are supposed to know that some R-packages are hold by Rstudio, and some are developed by core personnel or personnel closely related to R, or some prolific developers. Naturally, the number of downloads of these packages will probably be higher. On the one hand, the R-packages developed by these experienced developers may be more mature and perfect, on the other hand, users are more inclined to use the packages from famous authors, which can be regarded as a kind of “good use” guarantee.

However, the existence of these packages will make the results biased. Therefore, in order to explore the R-packages constructed by non-special creators, we have screened out four types of R-packages, namely:

- R-packages maintained by R studio
- R-packages created by authors from R core group
- R-packages created by authors from R secondary group
- R-packages created by R related authors
- R-packages created by top 20 prolific maintainers (This is resourced at *The most prolific package maintainers on CRAN* (n.d.))

And after that, we generated the lorenz curve here again in Figure 5.9, it can be observed that the distribution of R-packages in each download group is quite equal now and the Gini value decreases a lot as well, for the ‘extreme effect’ brought by highly-downloaded packages has disappeared. Thus, the user preferences could be shown more clearly. But as after filtering, the number of remaining packages is only 58, which is a too small-sized sample to construct representative conclusions, we would still focus on all the R-packages on CRAN in our later analysis.

Table 5.7: The number of downloads of different versions of R

R_version	count
3.2.1	4526
3.3.3	4305
3.1.2	3958
3.2.2	3781
3.4.0	3585
3.1.1	3458
3.5.1	3450
3.4.3	3409
3.4.1	3398
3.3.0	3362
3.4.4	3349
3.2.3	3308
3.0.3	3294
3.1.3	3274
3.3.2	3235
3.1.0	3216
3.5.0	3136
3.3.1	3107
2.15.3	2895
3.2.0	2873
3.4.2	2743
3.6.1	2636
3.0.1	2543
3.0.0	2537
3.0.2	2517
3.5.2	2514
3.5.3	2496
3.6.0	2374
2.15.0	2203
3.2.4	2122
2.15.1	2066
2.13.2	2011
3.2.5	1981
2.15.2	1906
devel	1856
3.6.2	1807
2.14.1	1791
2.11.1	1764
2.11.0	1733
2.14.2	1700
2.14.0	1660
2.13.0	1628
2.13.1	1585
3.6.3	1560
2.10.0	1496
2.10.1	1476
2.12.0	1432
2.12.2	1387
2.12.1	1362

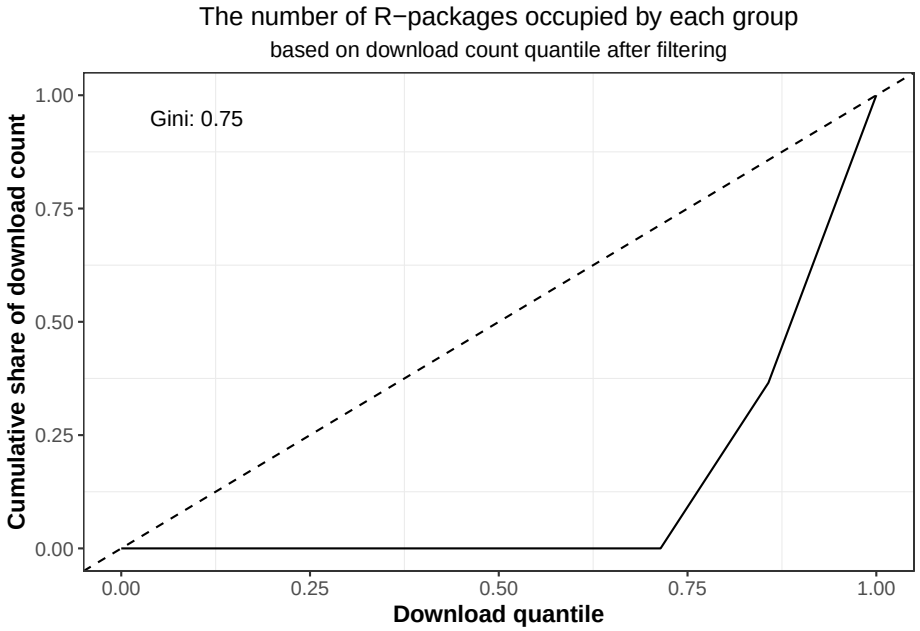


Figure 5.9: Percentiles of the download count against cumulative download count of R-packages after filtering four types of popular and highly-downloaded packages.

Table 5.8: Changed top 15 downloaded R-packages from 2013 to 2019

package14_13	package15_14	package16_15	package17_16	package18_17	package19_18
ncdf	XLConnectJars	reports	lava.tobit	ReporteRs	freetypeharfbu
playwith	KoNLP	moonsun	rggobi	OceanView	replyr
DMwR	doRedis	rPython	alr3	ReporteRsjars	zipcode
latticist	ElemStatLearn	rmongodb	tnam	gWidgetsRGtk2	rmosek
bstats	testthatsomemore	maxent	SweaveListingUtils	d3heatmap	msgpack
geoRglm	adehabitat	SDMTools	ElemStatLearn	-	-
reports	wmtsa	MSBVAR	zipcode	-	-
-	mixOmics	d3heatmap	-	-	-
-	DatABEL	ReporteRs	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-

Finding 1: The topic of newly added R-packages each year come from quite different areas.

Table 5.8 shows the R-packages that newly come up to the top 15 list each year, from which we can know how the user preference has changed year by year, compared with the previous year. To be more specific :

- For 2014, `ncdf` ranks first, which is used to provide an interface to netCDF format data.
- For 2015, the first downloaded is `XLConnectJars` on JAVA dependency.
- For 2016, the first one is `reports` for standardising the output of R.
- For 2017, `lava.tobit` aims to help with estimation and simulation of latent variable models.
- For 2018, the ranking-first `ReportsRs` is for creating Microsoft Word and Powerpoint documents.
- For 2019, `CALIBERrfimpute` is used to impute missing values in analysis datasets using full conditional specifications.
- For 2020, `SparkR` is similar to `dplyr` but for large datasets.
- For 2021, `heatmap.plus` is an extension of `heatmap()` function.

Finding 2: The topic of R-packages changes least between 2017 and 2019. And the most popular filed is about JAVA dependency.

Table 5.9 shows the R-packages that remain unchanged each year

Table 5.9: Unchanged top 15 downloaded R-packages from 2013 to 2019

package14_13	package15_14	package16_15	package17_16	package18_17	packag
Defaults	ncdf	XLConnectJars	XLConnectJars	XLConnectJars	XLCon
RSQLite.extfuns	epicalc	KoNLP	KoNLP	DMwR	KoNLP
gWidgetsRGtk2	gWidgets	DMwR	DMwR	KoNLP	DMwR
gWidgets	gWidgetsRGtk2	gWidgets	reports	SDMTools	SDMTo
alr3	DMwR	mixOmics	maxent	reports	gWidg
epicalc	alr3	gWidgetsRGtk2	SDMTools	mixOmics	ElemS
rggobi	-	-	mixOmics	ElemStatLearn	reports
its	-	-	gWidgets	alr3	gWidg
-	-	-	-	maxent	d3heat
-	-	-	-	gWidgets	alr3

compared with the previous year, from which we can know which packages are relatively stable in popularity.

To be more specific, packages like `XLConnectJars`, `DMwR`, `KoNLP`, and `gWidgets` are relatively popular through years from 2014 to 2021. And they are used for providing JAVA dependency, data mining, linguistic research and providing API for building toolkit-independent, interactive GUIs.

Let's take a look at the trending R-packages. Trending R-packages are ones that are downloaded at least 1000 times last week, which have increased significantly compared to the average weekly downloads in the previous 24 weeks(R-Hub n.d.[b]). That is to say, they are packages with high download volume in a recent short time. Through their topic, we can know what areas of R-packages people are concerned about recently.

Finding 3: The most popular topic of trending R-packages is for `r pkg_trendcom_count$topic[1]`.

Table 5.10 shows the first 15 trending R-packages along with their topics.

And Table 5.11 shows the first 10 topics for trending R-packages. The most popular topic is Bayesian, followed by Cluster and Econometrics. This is easy to understand, because the number of users from different fields is different, so there are also differences in the number of R-packages used in different topics. And it seems that Bayesian is quite a hit recently.

So, apart from the topic of the R-package itself, what other factors will probably be linked to its download volume? With this question in mind, we mainly explored the relationship between the total number of download counts of R-packages in the past year and the earliest release date, the

Table 5.10: Trending R-packages with topics

package	topic
mgcv	Bayesian
units	ChemPhys
metafor	ClinicalTrials
clusterSim	Cluster
fastcluster	Cluster
lhs	Distributions
mgcv	Econometrics
mgcv	Environmetrics
DiceDesign	ExperimentalDesign
lhs	ExperimentalDesign
Cubist	MachineLearning
metafor	MetaAnalysis
norm	MissingData
arulesSequences	ModelDeployment
clusterSim	Multivariate

Table 5.11: Topics of trending R-packages

topic	download times
WebTechnologies	5
Multivariate	4
MachineLearning	3
SocialSciences	3
TimeSeries	3
Cluster	2
Econometrics	2
MissingData	2
OfficialStatistics	2
Optimization	2

relationship between the last half a year's total download volume and the number of updates, the number of commits on GitHub repositories, the length of the name and the alphabetical order of the first letter of the name in the next following sections.

5.4 Compare last year's downloads with the earliest release date

Finding: R-packages that are initially released earlier on CRAN tend to have a higher download count in the past year. This perhaps is because in earlier times, there were fewer R-packages in the same category, so users had 'no choice' but to use them. Due to this, these R-packages accumulate a certain user base, which makes it more possible to attract new users.

In our common cognition, we may assume that the earlier a R-package is released, the more people will know about it, and thus the more downloads it will have. However, R-packages related to different topics cannot be directly compared, because it is possible that the total download amount of R-packages in a certain topic is higher than that in another topic. Therefore, in order to test this conjecture as clearly as possible, we selected three domain R-packages through CRAN task view(n.d.), calculated their respective downloads in the previous half a year, and extracted their earliest release dates for comparison. Those three topics are :

- a) R-packages for Time Series Analysis

The first topic is Time Series Analysis. Time Series Analysis is a statistical technique that deals with time series data, or trend analysis. Time series data means that data is in a series of particular time periods or intervals(*Time Series Analysis* 2020).

- b) Bayesian R-packages for general model fitting

The second topic is Bayesian Inference. Bayesian statistics is a mathematical procedure that applies probabilities to statistical problems. It provides people the tools to update their beliefs in the evidence of new data(perpetual 2019).

- c) Econometrics R-packages

In order to test whether this is the case in other areas, the last topic is for econometrics R-packages. Econometrics is the use of statistical methods using quantitative data to develop theories or test existing hypotheses in

economics or finance, which relies on techniques such as regression models and null hypothesis testing(Hayes 2020).

Figure 5.10 displays the scatterplot of the past year's download count and the earliest release date for `Time Series Analysis`, `Econometrics` and `Bayesian` R-packages. It can be seen that generally, as the earliest release date gets later and later, the number of download logs becomes lower and lower. And for `Time Series Analysis` R-packages, they are mainly released between 2012 and 2019. For `Bayesian` R-packages, most of the R-packages are from 2007 to 2012. And most `Econometrics` are centered between 2013 and 2016.

In conclusion, we are not surprised to find that the earlier the R-package is released, the more downloads it could has, which is reflected in all of three topics of R-packages above. That is probably because the R-packages released earlier will be better known. When they are released early, there may be a relatively small number of R-packages of the same topic, under non-serious competition. As a result, the R-packages coming later can easily be covered up, since people generally tend to use well-known, mature and habitual packages.

That is to say, earlier R-packages are more conducive to the cultivation of user habits. After all, habits are influenced by the length of time. For example, if the teacher is an old user of some R-packages, they may recommend these R-packages to their students when they teach, or colleagues may prefer to recommend familiar R-packages to others especially when they get a satisfying user experience.

5.5 Compare moving average of fable and forecast

Finding: R-package 'forecast' has a more stable download trend comparing to 'fable'. And when 'fable' gets updated, its downloads peaked, while 'forecast' suffers a dropping on the contrast.

As we stated in the previous section, the earlier the R-package is released, the easier it is to get a relatively higher total download. But that doesn't mean a better growth will exist. Due to that, in this section, we compared package `fable` and `forecast`. They are two closely related R-packages, for `fable` is the later released tidy version of `forecast`. And in this way, we can approximately fix all the factors except the initial release date, so that we can compare the growth and changes of these two more clearly.

Figure 5.11 and Figure 5.12 show the daily download count changing

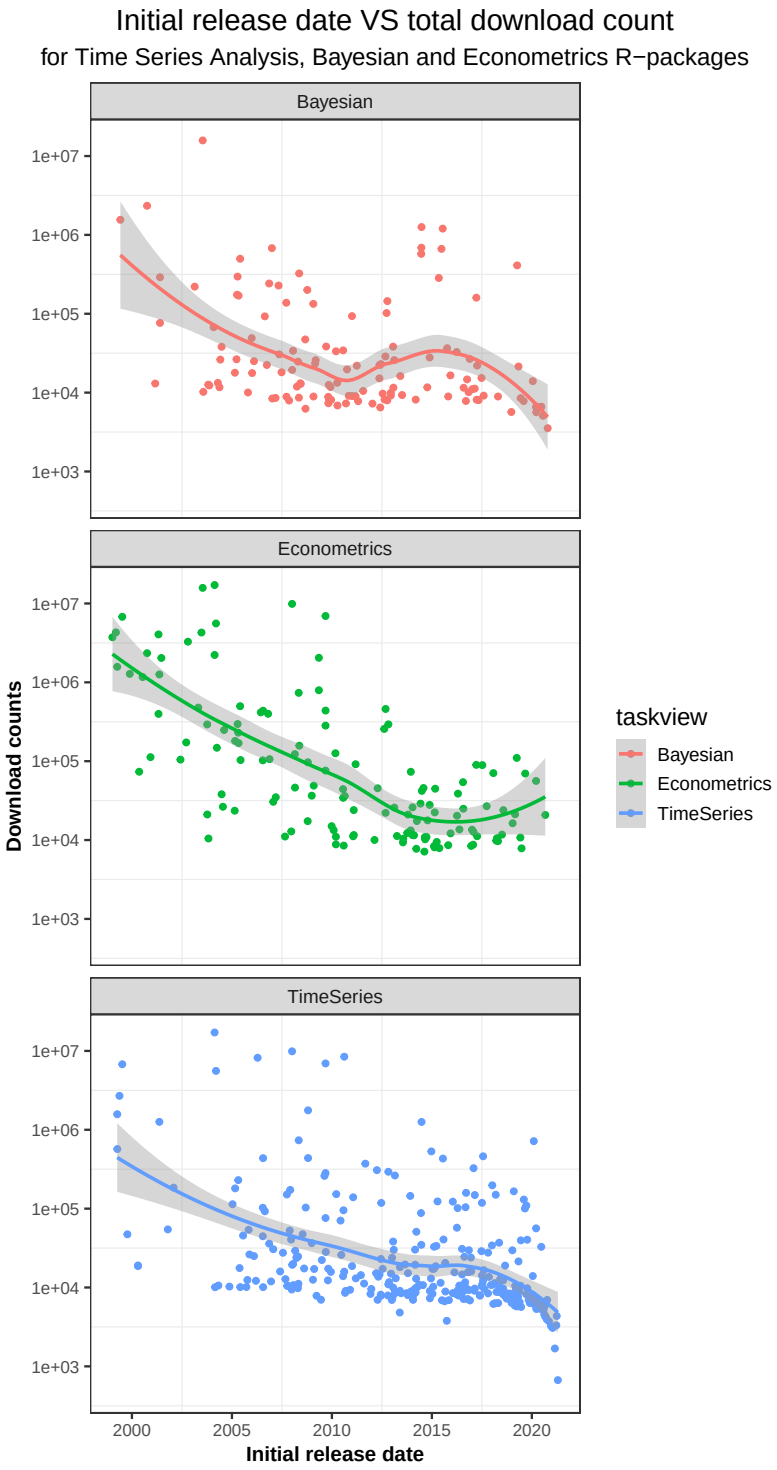


Figure 5.10: The download count decrease with the initial release date.

during last half a year, which indicates strong weekly seasonality. And that means the downloads tend to be higher in week days and thus lower on weekends, which is consistent with total R-package trend analyzed before.

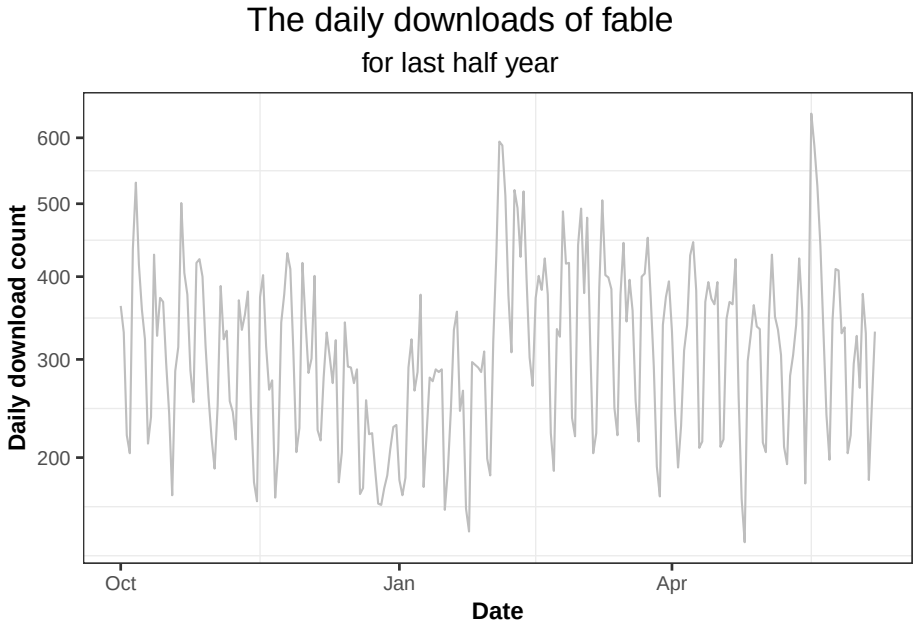


Figure 5.11: The daily download of R-package “fable”

Therefore, in order to estimate the trend-cycle and reduce the weekly seasonality to see the changes more clearly, we introduced the Moving Average (MA).

A moving average of order **m** can be written as :

$$T_t = \frac{1}{m} \sum_{j=k}^{k} y_{t+j}$$

where **m=2k+1**. That is, the estimate of the trend-cycle at time **t** is obtained by averaging values of the time series within **k** periods of **t**.

And here, we considered the equal weighed 7 moving average. That is, it calculates the weighted average for every seven consecutive time series with the following weights : *[1/7,1/7,1/7,1/7,1/7,1/7,1/7]*.

Figure 5.13 shows the moving average (MA) of `fable` and `forecast` respectively. They have quite different moving average patterns with `forecast`’s download volume much higher than `fable`’s. That is, the MA of

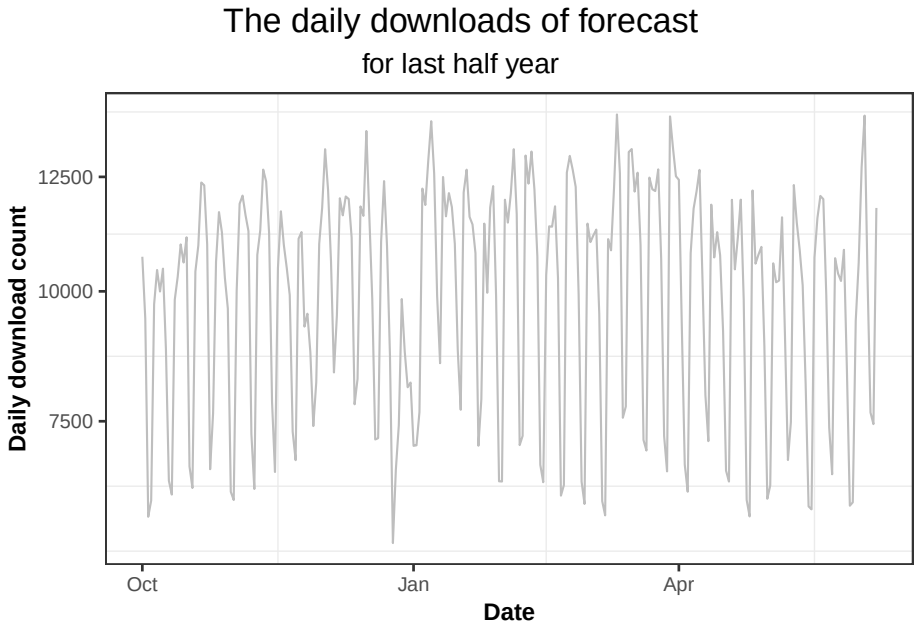


Figure 5.12: The daily download of R-package “forecast”.

`forecast` is relatively stable than that of `fable` except for the time around New Year’s Eve when `forecast` has a significant drop. But during that time, a drop also appears in `fable`, which was probably due to the big New Year holiday. In addition, the purple vertical dashed line in plot of `fable` marks the update day of it which is on 2021-1-29. And soon after that day, its downloads peaked, which was because the number of download counts will increase on the update day. At the same time, `forecast` drops and then only gets a gentle increase.

In conclusion, the download variance of `forecast` is larger than that of `fable` while the former looks relatively more stable than the latter after applying the moving average. And once again, it coincides with the conclusion of the previous section, that is, the earlier released R-package tends to gain more cumulative downloads. In addition, we can also see that the growth of both of them sometimes increases at the same time, while sometimes the growth of one corresponds to the decline of the other. As for as we concerned, the shared growth may be due to their similarity. Users may download both at the same time or compare them like us. And the trade-off may also be due to the similarity of their functions, one can be replaced by the other in the process of using. Finally, we can also know that earlier release does not guarantee a faster growth. For example, `fable` has a more dramatic growth than `forecast` from 2021-01-21 to

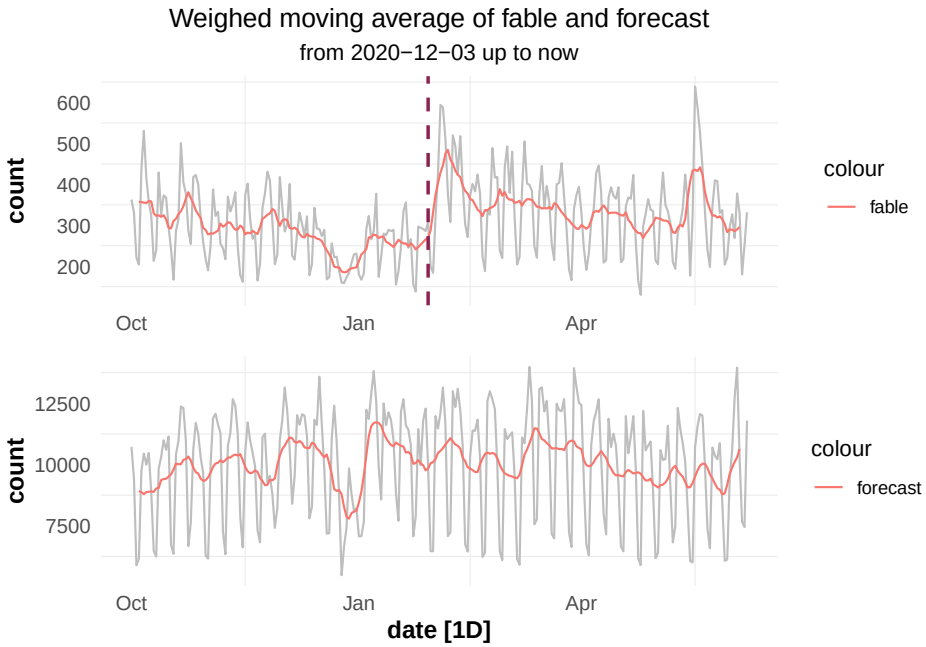


Figure 5.13: The moving average of R-package “forecast” and “fable”.

2021-02-06.

5.6 Compare download counts with the number of commits on master branch

In this section, we compared last half a year’s total downloads with the number of commits on master branch in Github repositories. And this turn, we initially planned to replace the research object with all the R-packages on CRAN, 17,699 up to now. But we soon found out that there were only 6185 R-packages that have a Github repository, and after cleaning up, only 5769 remained.

Finding: In our initial assumption, more commits on master branch of Github repository tends to bring more download counts to R-packages. On the one hand, the number of commits indicates that developers are constantly supplementing and updating the R-package, which will lead to more downloads. On the other hand, the number of commits also reflects the attention developers attach to the R-package to a certain extent. So, if an R-package has more commits, then developers may invest more in advertising and other ways to promote it, so as to expand the popularity and improve the download count.

The way we initially applied was extracting the commits through accessing the Github API. To achieve this, we first scraped the Github URLs from 'description' page for all the R-packages, and cleaned up the multiple URLs or some other redundant symbols and characters. And the URL of scraping the commits through Github API is formatted as *"https://api.github.com/repos/%7Buser_repo%7D/commits?per_page=1"*. For example, the URL for package `tidyverse` can be like *"https://api.github.com/repos/tidyverse/tidyverse/commits?per_page=1"*. So, practically, by replacing the `user_repo` part that consists of the name of the related repository and the name of the package holder, we could access the content of all the package URLs through Github API.

Based on that, Table 5.6 shows the first 5 R-packages of the whole, their Github URLs and the `user_repo` part.

However, the Github API has a rate limits allowing for up to 60 requests per hour for unauthenticated requests, and this can be extended to 5000 per hour after authentication(*Resources in the REST API* n.d.). But even after getting authentication, our rate limit didn't get promoted. So, we switched to scrape commits by python spider. And by setting random user agent, we successfully avoided the API limit. And as this method is quite time consuming (around 5 hours), we saved the output as a .txt file then load it in R studio to speed up the compiling process.

\begin{table}

\caption{First 5 R-packages with their 'user_repo' element}

Table 5.12: First 5 R-packages with commits on Github and last half year's total downloads

package	commits	total
rlang	4,427	14748569
dplyr	7150	12054840
ggplot2	4,756	11825489
lifecycle	206	10438007
tibble	4,286	10340887

package	URL	user_repo
abbyyR	http://github.com/soodoku/abbyyR	soodoku/abbyyR
ABCoptim	http://github.com/gvegayon/ABCoptim	gvegayon/ABCoptim
abctools	http://github.com/dennisprangle/abctools	dennisprangle/abctools
abdiv	http://github.com/kylebittinger/abdiv	kylebittinger/abdiv
abess	http://github.com/abess-team/abess	abess-team/abess

\end{table}

Table 5.12 shows the first 5 R-packages of all, along with their last half a year's total downloads and the number of commits on Github master branch.

Figure 5.14 shows the scatterplot along with a smoothing line. In general, more commits, more downloads.

Another method to explore this question is by looking at the ultra-low-downloaded R-packages, whose download count only ranks at last 1% of all. And another purpose for this part is to show how we initially intended to scrape commits through Github API with R (as the sample size is less than 60 R-packages under this situation).

From Table 5.13 we could see the last 1% downloaded count is around 401. And as these download counts are extremely low, we could assume that many factors will have little effect on their downloads. Thus, we can further assume that the only two differences between them are the number of commits on Github master branch and the total download count.

So, we could filter these ultra-low-downloaded R-packages and extract the commits on Github, then explore the pattern in total download count against the number of commits in Figure 5.15. It could be believed that when the number of commits increases from 0 to 100, the download count first increases and then decreases. After that, the download volume keeps rising and appears to have a little jump at the end, but we supposed it might be due to the the too small-sized sample, and the observation

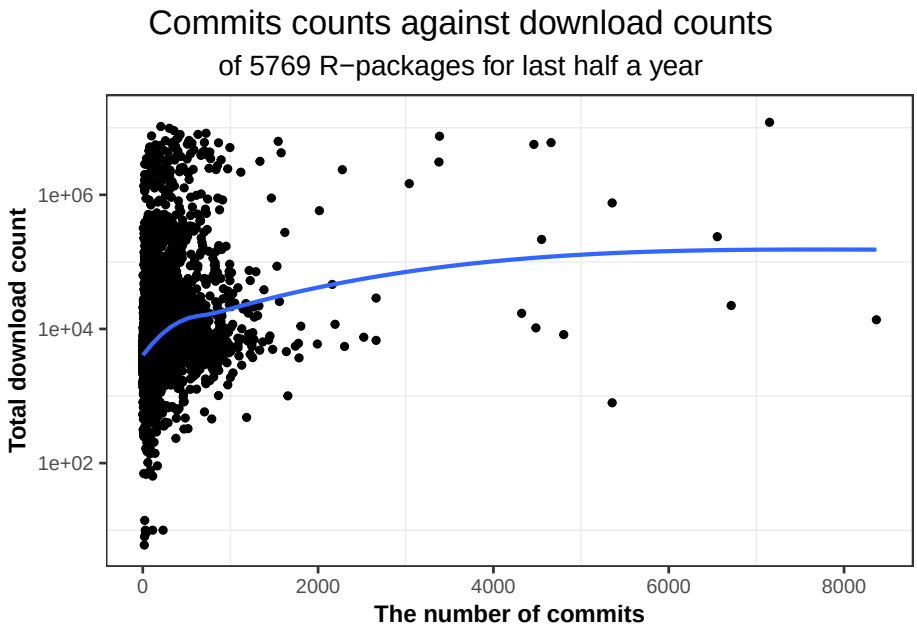


Figure 5.14: The commits on master branch of Github repository against the last half a year's total download count.

causing the decline may be an outlier. So, if expanding the observation horizon, we expected the curve might increase again. Therefore, it can be seen that even for these last 1% downloaded R-packages, there also exist the phenomenon - "the more commits, the more downloads".

5.7 Compare download counts with the number of updates

Next, let's look at the relationship between the number of updates and the number of downloads. In this part, our analysis object is still all the packages.

Table 5.13: Quantile of total download count for all the R-packages on CRAN

	download count
0%	0.00
1%	373.00
2%	716.00
3%	1087.65
4%	1365.20
5%	1667.50
6%	1915.30
7%	2185.85
8%	2334.00
9%	2512.00
10%	2686.50
11%	2796.00
12%	2862.60
13%	2913.00
14%	2967.00
15%	3027.00
16%	3087.00
17%	3156.00
18%	3245.00
19%	3333.00
20%	3406.00
21%	3466.00
22%	3532.00
23%	3581.00
24%	3627.00
25%	3664.00
26%	3700.00
27%	3731.85
28%	3766.00
29%	3808.00
30%	3846.00
31%	3884.00
32%	3929.00
33%	3973.15
34%	4016.00
35%	4062.00
36%	4101.00
37%	4141.00
38%	4187.00
39%	4227.00
40%	4269.00
41%	4310.00
42%	4352.00
43%	4391.00
44%	4434.20
45%	4477.75
46%	4527.00
47%	4568.00
48%	4617.00

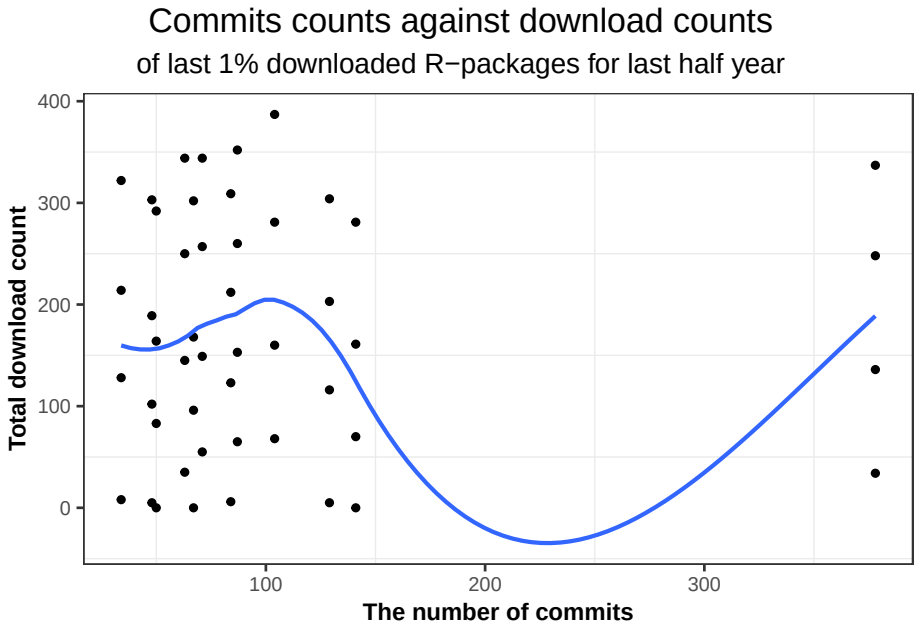


Figure 5.15: The commits on master branch of Github repository against the last half a year's total download count for last 1% downloaded R-packages on CRAN.

Finding 1: The download count tends to rise with the number of updates.This is because as we have known before, when a package is updated, there will be a significant increase in the number of downloads. That is to say, when a package is updated, not only old users will download the latest version, but also it is easier to attract new users at this time, for the increase of downloads in a short time may bring it into the trending list.

Figure 5.16 shows that the number of downloads increases with the update times. And most R-packages are updated no more than 30 times.

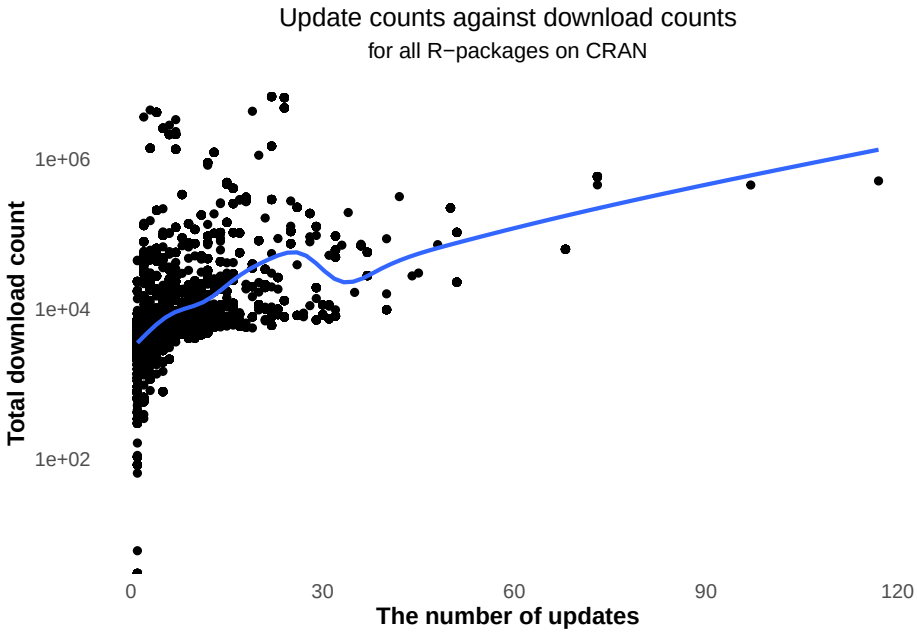


Figure 5.16: The download count increase with the number of updates.

Finding 2: Over half of the R-packages don't tend to update very frequently.

Also checking Table 5.14, we can know that the percentage of R-packages whose updates are less than average is 70.2920266 %, which means much more than half of the R-packages do not tend to update very frequently.

Table 5.14: Percentage of trending R-packages whose updates are less than average

number of packages with low updates	percentage of packages with low updates
11963	70.29203

Finding 3: Most R-packages keep updating with the time to keep its activity.

It can be seen from Figure 5.17 that most R-packages' latest publish date is after 2015, which indicates that many R-packages tend to update by time to keep its activity.

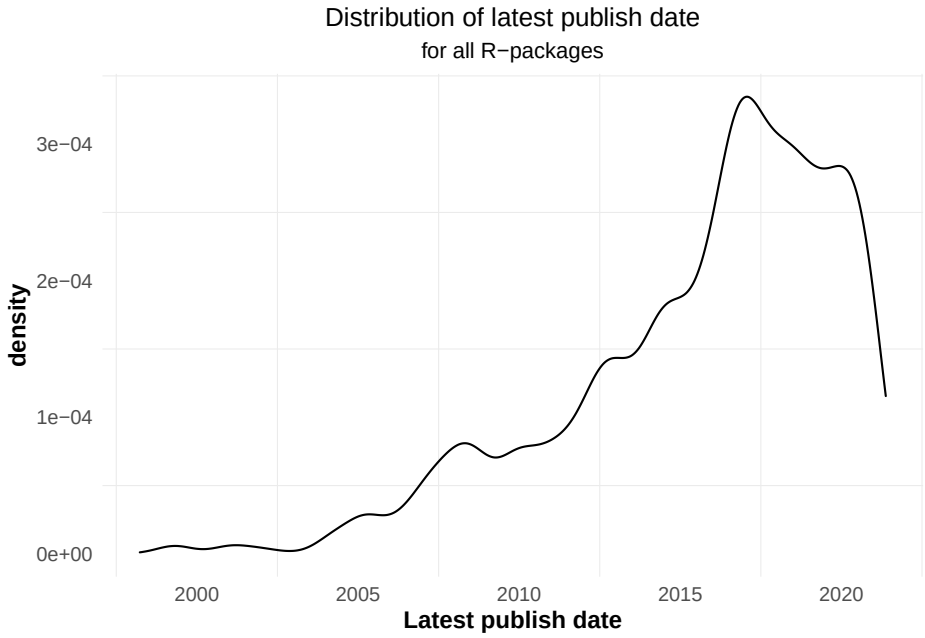


Figure 5.17: The latest update dates of trending R-packages are almost all the recent dates.

Finding 4: Most of the R-packages are likely to update at a longer time interval.

Figure 5.18 shows that with the increase of update interval, the number of downloads first increases and then decreases slightly. And most of the time

intervals are between 45 and 450 days, which shows that their update frequency is not very high.

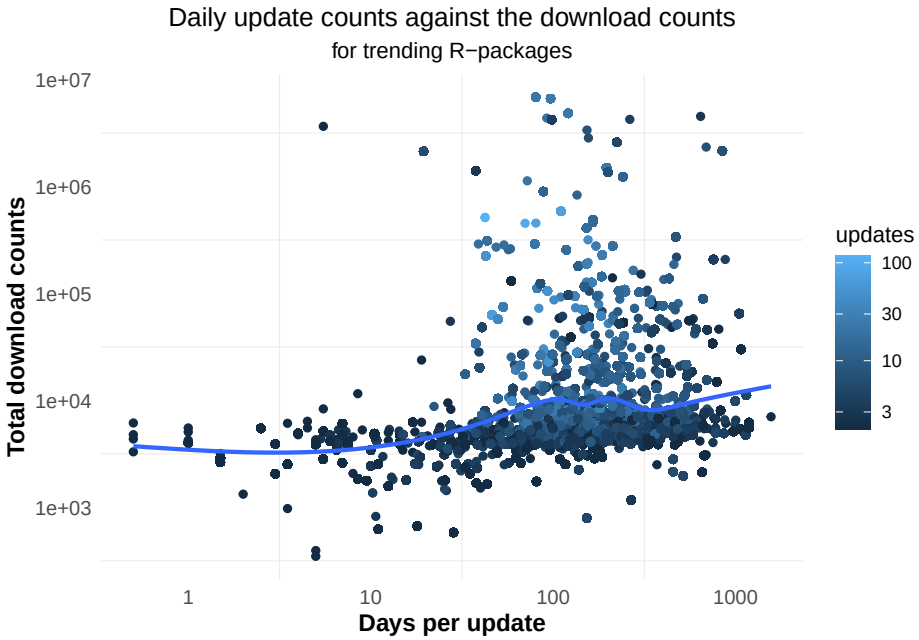


Figure 5.18: Most of the time intervals of updates are between 45 and 450 days.

In conclusion, it's not that the more updates the R-package has, the popular it will be. In general, most of the R-packages whose updates are lower than the average occupy the majority. Therefore, we can also know that the total number of updates is not very important for the R-package. The important thing is to keep it updated with the time.

5.8 Compare the package name length with download counts

Here, we still focused on all R-packages on CRAN and made comparison between the name length with the last half a year's total downloads.

Finding 1: The name length of R-packages has no significant effect on the number of total downloads for last half a year.

We could see from Figure 5.19 that, in general, the influence of name

length on download volume is not very obvious. But we could still observe that the number of downloads decreases slightly with the length of the name. And the names with more than 6,000,000 downloads are between 5 and 9 characters long. Also, the most downloaded is R-package rlang whose name length is 5, with 1.5572507×10^7 download counts.

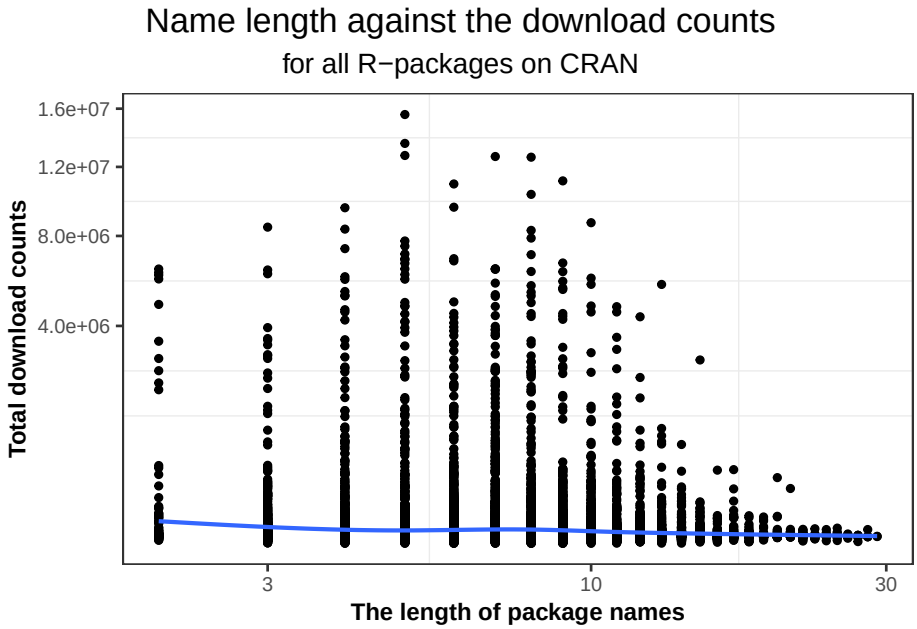


Figure 5.19: The names of R-packages with more than 6,000,000 downloads are between 5 and 9 characters long.

Finding 2: The average name length of R-packages is about 7.8 characters, and over half of the R-packages tend to have shorter names, which may make it more easier to be remembered by users.

Table 5.15 shows that the average name length all R-packages is 7.8461146. And over half of the R-packages are more likely to have names shorter than average. That means, to a certain extent, R-packages with shorter names are easier to get relatively higher downloads. And that may because shorter named packages are easier for users to remember.

Finding 3: The 95% confidence interval of the name length is between 7.80 and 7.90 characters.

Table 5.15 shows the 95% confidence interval of name length for all

Table 5.15: Percentage of packages whose name length are less than average

number of short names	percentage of short names	total average name length
9286	52.59402	7.846115

packages. We could know that the 95% suitable name length is between 7.7936087 and 7.8986206 characters.

\begin{table}

\caption{95% Confidence Interval of name length for random packages}

lower.bound	upper.bound
7.793609	7.898621

\end{table}

After we found that there is no obvious relationship between the name length of the package and the download volume, a new question came up : Is the name length of the package affected by the time of initial release date?

Finding 4: For taskview R-packages, the length of the package name increases with the initial release date, especially for Bayesian packages.

We have such an experience in life, that is, for the same type of goods, such as detective novels, the later they are released, the narrower the choice of naming is, because there are certain restrictions on the name of a particular type, but many names have been occupied by the books published earlier. Therefore, these later published books often have to lengthen their names to distinguish themselves from the existing books in the same type or even the same name. Coincidentally, we assumed the naming of R-packages from the same topic would also be affected by the initial release time, to a certain extent. So, here, we looked back to the CRAN task view R-packages for constructing comparison among R-packages from the same topic. Figure 5.20 shows the name length of CRAN task view R-packages against the initial release date. It is obvious that the length of the R-package name increases with the initial release date, especially for Bayesian R-packages.

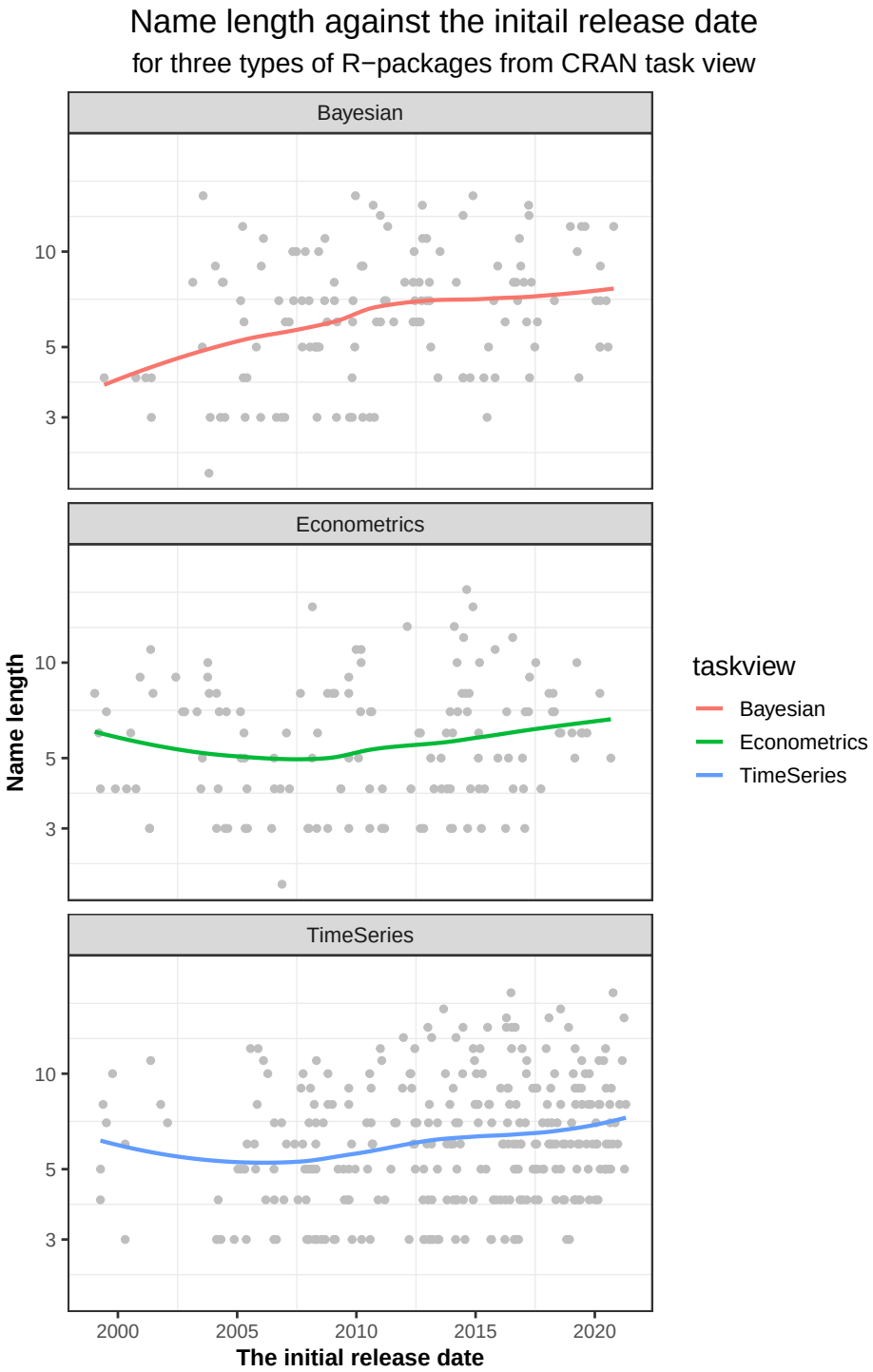


Figure 5.20: The name length of taskview R-packages slightly increase with the initial release date.

Finding 5: For all R-packages on CRAN, the average name length tends to generally increase with the initial release date.

Although we'd better explore this question among the same topic of R-packages, we could also have a view on the annual change in the average name length of all R-packages on CRAN.

Figure 5.21 shows the average name length of all R-packages on CRAN released in each year. It is obvious that the name length of these R-package generally increases year by year.

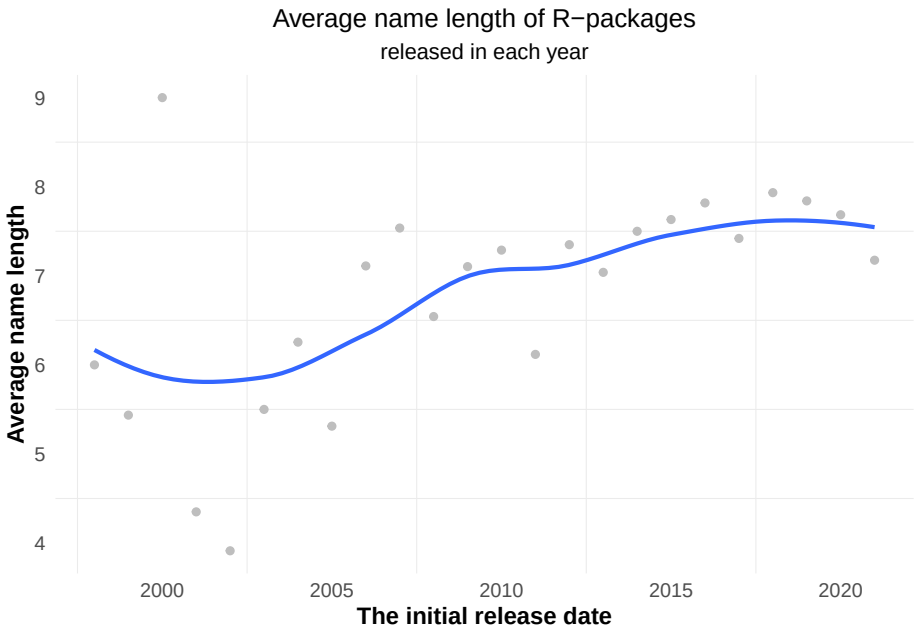


Figure 5.21: (ref:avgnamelth-year)

5.9 Compare download counts with alphabetical order of name

We might also have such experiences in our life. For example, when we go shopping in the supermarket, the goods placed in front of the shelves will be easier to attract our attention and be selected by us, because we may no longer want to seek products with similar functions for a long time.

From the perspective of R-package name, in addition to the length, is the alphabetical order of the first letter also linked the download volume? For

R-packages with earlier alphabetical order will be placed at the first part of the R-package list on CRAN(*Available CRAN Packages By Name* n.d.). To answer this question, we grouped the R-packages by 26-letter order, calculated the average downloads of each group, and then made comparison.

Finding 1: For all R-packages, the average downloads of different alphabetical group are slightly increasing by its order, while the total download tends to decrease a little, instead.

From Figure 5.22 we could see that the average downloads of different alphabetical group are slightly increasing by alphabetical order, while the total downloads tends to decrease a little, instead. This is because the later-ordered group contains fewer R-packages. It can be seen that developers may prefer to name their packages with a top alphabetical order, which might be easier for users to notice.

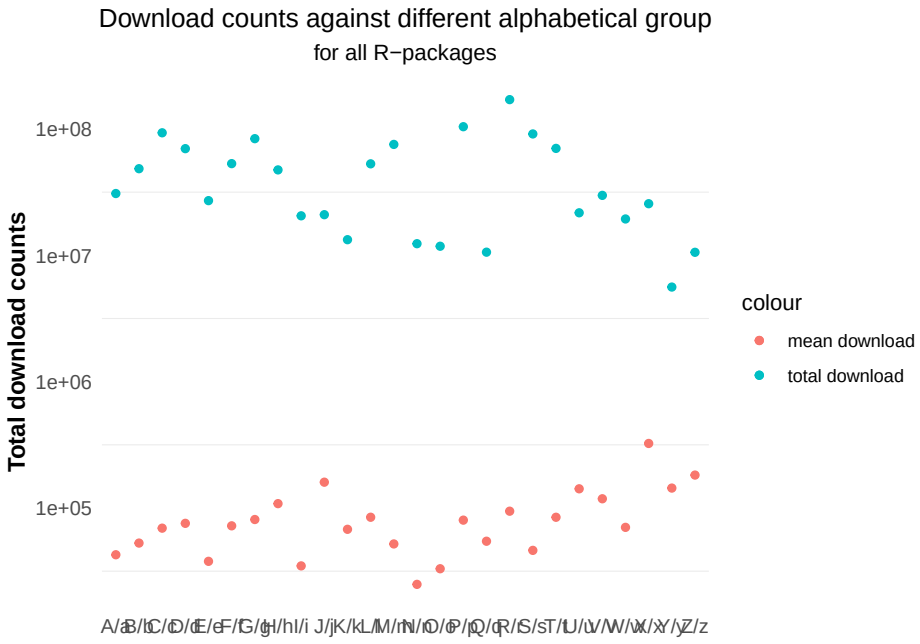


Figure 5.22: The average total download count of each group is little linked to the alphabetical order of R-package name.

Finding 2: For all R-packages on CRAN, the group with higher total download tends to have greater variance, owing to more outliers.

Then, let's take a look for how the variance varies across groups. Figure 5.23 shows the data range and the median value for each alphabetical group. It can be seen that the group "R/r" has the highest outlier and the group "X/x" has the largest variation. At the same time, we could also see that the variance between different groups is not very obvious, which means that for each group, 50% of the R-package download count is relatively concentrated. The real difference is the highest and lowest downloads per group. In general, the larger the total number of downloads (which also means the more packages in the group), the more outliers will be included, such as group "F/f", "L/l" and "R/r".

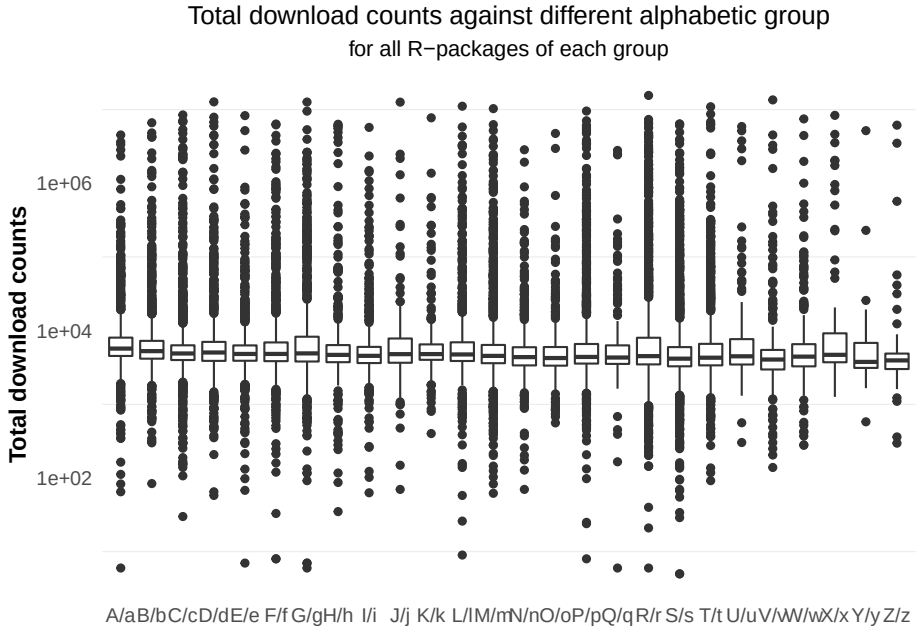


Figure 5.23: The R-packages with name starting with "j" has the largest variation.

In order to further verify our conclusion, we turned to the ultra-low-downloaded R-packages. As we have mentioned previously, when it comes to the ultra-low-downloaded R-packages, we could approximately assume the only variable that may affect the amount of downloads is name order here. From Figure 5.24, we could see that the difference in median download count of each alphabetical group is not significant as we expected.

Therefore, we could approximately draw a conclusion : In general, the R-packages with the top alphabetical order are easier to get relatively high download volume, but the gap is not significant. At the same time, the

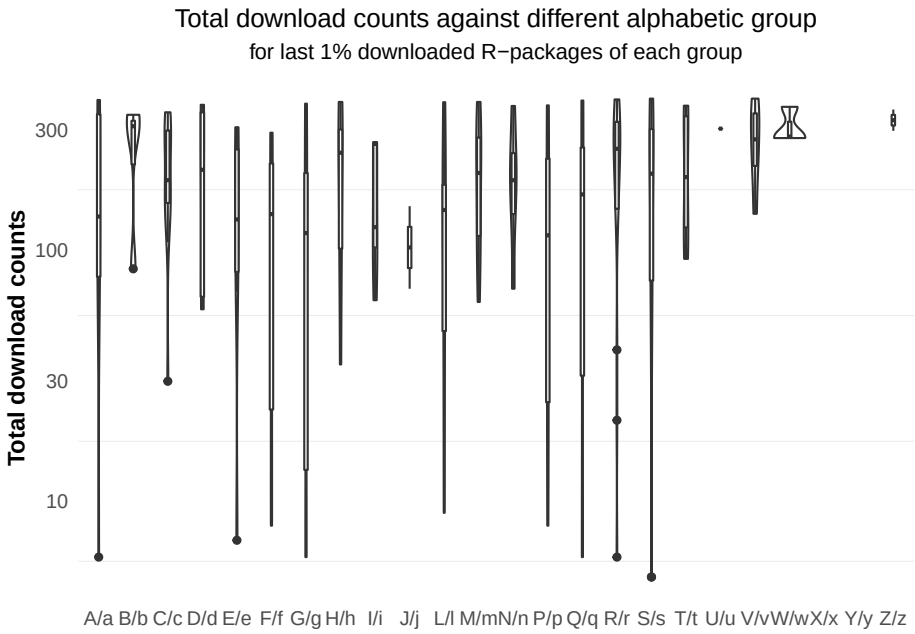


Figure 5.24: (ref:random-boxplot)

higher the number of downloads, the greater the variance will appears in this group.

Chapter 6

Summary

In this project, we collected summary daily download logs of R-packages through web Application Programming Interface (API) maintained by r-hub(R-Hub n.d.[a]) and also used daily download data in CRAN for a time period from 2013-04-01 to 2021-04-01 to explore the daily download pattern of all R-packages both in general and of each year. In that case, we found that it is true that the cumulative number of downloads increases with time, and the variance also increases, which indicates that some R-packages with larger downloads grow rapidly. In addition, there is also strong weekly seasonality in the daily download plot. The download count will peak through weekdays and drop on weekend, for users may tend to work and study during weekdays while rest on weekends. What's more, through Lorenz curve, we also found that most of the cumulative downloads came from the top 10% downloaded R-packages, so we could also see that the distribution of downloads is quite unequal. Part of the reason is that these top 10% downloaded R-packages contain quite a lot popular and frequently used R-packages, such as `tidyverse` and `rlang`, which would be more probably to gain high downloads. In addition, there are other R-packages that often get high download volume, which can be divided into the following four categories:

- R-packages maintained by R studio
- R-packages created by authors from R core group
- R-packages created by authors from R secondary group
- R-packages created by R related authors
- R-packages created by top 20 prolific maintainers (This is resourced at *The most prolific package maintainers on CRAN* (n.d.))

However, the existence of these R-packages may make it difficult to reflect the popularity of other R-packages, so we excluded these R-packages for

the analysis of user preferences. And we found that the topic of newly added R-packages on 1st Oct of each year are from quite different areas, while the R-packages remaining most stably popular during 2017 to 2019 is about JAVA dependency. Definitely, JAVA always ranked the top three among programming languages according to TIOBE Index(*Latest news* n.d.), which shows that the number of users under JAVA related R-packages would be probably larger.

As for R itself, its download pattern is quite similar to that of total R-packages on CRAN. And the most used OS for R users is windows OS. Also, the most popular version of R is 3.2.1.

After exploration of the characteristics of download pattern for R-package and R itself, we then extracted the release dates of all R-packages and task view R-packages from CRAN to compare the total download count (past year for task view R-packages and last half a year for all CRAN R-packages) among R-packages with different release date or with different numbers of updates. And we found that for R-packages from the same topic, earlier release date usually would bring more download count, while R-packages with more times of updates would not always have higher downloads. So to sum up, R-packages released earlier and kept being updated are more likely to have higher downloads.

In the next section, we initially tried to scrape the number of commits in Github repository of all available R-packages on CRAN through Github API by R, to check whether more commits would result in more downloads or not. But there came a tricky problem on the rate limit of Github API. As documented in *Resources in the REST API* (n.d.), unauthenticated users could only be permitted to send 60 requests per hour. And only after get authentication, could the rate limit be expanded up to 5000 per hour. However, after trying several methods to get authentication, the rate limit was failed to be increased. So, we switched to make this done with python by setting random user agent to avoid the API limit. Meanwhile, in order to display our initial research idea, we still had a look at the last 1% downloaded R-packages on this question with the original method, and that is also adapted as well. Therefore, we would expect that generally, if an R-package has more commits on Github repository, it would probably gain more download count.

The last two parts are about analysis for R-package name. We compared the average downloads among R-packages with different name length and different alphabetical orders. It is believed that over half of the R-packages tend to have shorter names probably for the sake of being easily remembered by users. And alphabetical order played little roles in promoting the download volume.

Chapter 7

Conclusion

In conclusion, we hold the belief that there are many factors that could be linked to the download amount and popularity of the R-packages on CRAN, such as the popularity of the creator, the application field of the R-package, the release date, whether to keep updated, and the length of the name. In addition, we also assume that the number of commits on Github repository may also probably have some correlations with the download amount of R-packages. Anyway, we could generally believed that a relatively popular R-package should have earlier release date, shorter name and maybe more commits on Github repository if they have. There is also another point that could not be ignored is that it is better to keep updated.

Bibliography

(N.d.). URL: https://cran.r-project.org/.

(N.d.). URL: https://cran.r-project.org/web/views/.

Available CRAN Packages By Name (n.d.). URL: https://cran.r-project.org/web/packages/available_packages_by_name.html.

CRAN Mirrors (n.d.). URL: https://cran.r-project.org/mirrors.html.

Csárdi, Gábor (2019). *cranlogs: Download Logs from the 'RStudio' 'CRAN' Mirror*. R package version 2.1.1. URL: https://CRAN.R-project.org/package=cranlogs.

Csárdi, Gábor and Maëlle Salmon (2020). *pkgsearch: Search and Query CRAN R Packages*. R package version 3.0.3. URL: https://CRAN.R-project.org/package=pkgsearch.

Forecasting: Principles and Practice (3rd ed) (n.d.). URL: https://otexts.com/fpp3/stl.html.

Hayes, Adam (Dec. 2020). *Econometrics: What It Means, and How It's Used*. URL: https://www.investopedia.com/terms/e/econometrics.asp.

Ihaka, Ross (1998). *R : Past and Future History. A Draft of a Paper for Interface 98*. URL: https://www.stat.auckland.ac.nz/~ihaka/downloads/Interface98.pdf.

Latest news (n.d.). URL: https://www.tiobe.com/tiobe-index/.

Morgan-Wall, Tyler (2017). *adjustedcranlogs: Remove Automated and Repeated Downloads from 'RStudio' 'CRAN' Download Logs*. R package version 0.2.0. URL: https://github.com/tylermorganwall/adjustedcranlogs.

perpetual, NSSI am a (June 2019). *Bayesian Statistics Explained in Simple English For Beginners*. URL: https://www.analyticsvidhya.com/blog/2016/06/bayesian-statistics-beginners-simple-english/.

Peter Li (2021). *packageRank: Computation and Visualization of Package Download Counts and Percentiles*. R package version 0.4.2. URL: https://CRAN.R-project.org/package=packageRank.

Pettinger, Tejvan (n.d.). *Lorenz Curve*. URL: https://www.economicshelp.org/blog/glossary/lorenz-curve/.

Ponce, Marcelo (2021). *Visualize.CRAN.Downloads: Visualize Downloads from 'CRAN' Packages*. R package version 1.0.1. URL: https://CRAN.R-project.org/package=Visualize.CRAN.Downloads.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

R-Hub (n.d.[a]). *r-hub/cranlogs.app*. URL: https://github.com/r-hub/cranlogs.app#the-api-of-the-cran-downloads-database.

– (n.d.[b]). *r-hub/cranlogs.app*. URL: https://github.com/r-hub/cranlogs.app.

Resources in the REST API (n.d.). URL: https://docs.github.com/en/rest/overview/resources-in-the-rest-api.

The GNU Operating System and the Free Software Movement (1991). URL: http://www.gnu.org/.

The most prolific package maintainers on CRAN (n.d.). URL: https://blog.revolutionanalytics.com/2018/03/the-most-prolific-package-maintainers-on-cran.html.

Time Series Analysis (June 2020). URL: https://www.statisticssolutions.com/time-series-analysis/.

Warren, Tom (Apr. 2017). *Apple reveals Windows 10 is four times more popular than the Mac*. URL: https://www.theverge.com/2017/4/4/15176766/apple-microsoft-windows-10-vs-mac-users-figures-stats.