What makes an R-package popular?

Yiwen Zhang

2021-06-14

Contents

Preface

As an R user, have you ever thought about what kind of information is hidden behind the numbers of downloads for R and R-packages? Why would there be an abnormal download volume? When choosing which package to use, we likely often tend to use popular ones. But what kind of packages are popular? And what factors can be linked to the popularity of an R-package? I propose those are questions as those most R users will be curious about. Therefore, based on the existing findings, this report attempts to provide answers and directions of enquiry towards the above questions, by taking an exploratory data analysis approach.

This report 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 Drs Emi Tanaka and Hien Nguyen for their generous help on writing. I would like to also thank Professor Rob J Hyndman, for holding meetings every two weeks 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 probably influential factors that drive such popularity. Data are closely related to social and economic activities in our daily life, so data mining and analysis can often be beneficial. This research focuses on R in this project, and aims to explore the popularity of R-packages from CRAN (the Comprehensive R Archive Network). To achieve this, it is assumed in this research that the download volume is a relatively reliable and simple measurement for the popularity of an 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 can be linked to the download volume. The technical aspects that are applied in this research is largely compromise of R programs and scripts, together with some Python code on data scraping. By obtaining the daily and recent half year total download counts of 17,713 R-packages and R itself, this research analyzes pattern, characteristics, as well as the relationship between the release dates, update times, numbers of commits on Github (master or main branch), name lengths and alphabetical order of the names. Finally, this research also provides some description of the changes of R-user preferences on 1st April over the period of 2013 to 2021.

Results show that the daily downloads of R-packages have a strong weekly seasonality and unusual spikes caused by repeat downloads, updates or server test issue. The most stably popular R-packages over the period of 2017 to 2019 is related to JAVA dependency. We also found that an earlier release date is more likely to result in a higher download volume for R-packages. Similarly, update times and commits numbers on master (main) branch in Github repositories are also positively correlated with download volumes. However, the alphabetical order of R-package names contributes

little to the download counts.

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

Chapter 2

Introduction

Data have penetrated into every industry and business field, has become an increasingly important factor of production and consumption in our lives. However, data must be processed to uncover the information within. For this reason many statistical tools have been produced, such as SPSS, Python, R and so on. In this project, we focus on the R language (**?**).

R is a programming 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 terms of the Free Software Foundation's GNU General Public License (**?**). R is one of the most popular statistical languages and has been ranked competitively, even among the general purpose programming languages (peaking 8th in August 2020 in **?**). 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 have a larger user base.

Today, R is greatly enhanced by over 17,713 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 (**?**), the contributions were managed by only a small group of core developers. In April of 1997, the 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 counts for each package is only readily available from the RStudio CRAN mirror. In this report, we will be focusing only on one CRAN mirror of RStudio, albeit it being the most popular one, for ex-

ploratory 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 (**?**).

The most intuitive measurement of popularity is the total number of downloads, although it does not directly reflect the number of users, because it includes repeated downloads by the same user, updates and test downloads caused by the server. Briefly, some works concerning the reporting and interpretation of these download counts include: the R-package `adjustedcranlogs` (**?**), which attempts to correct for download inflation using a heuristic approach; the package `packageRank` (**?**), which provides a way to compute the rank percentile for packages and filters invalid downloads, including small and medium size logs. There are also some packages such as `pkgsearch`(**?**) and `Visualize.CRAN.Downloads`(**?**) that provide some visualization methods to explore the package download trends or for convenient searching. More details are provided in Section 5.

Based on those initial works, researchers have started to study the popularity of R-packages from CRAN. We still assume in this project that, the download amount is a relatively reliable and simple measure of a R-package's popularity. The previous studies, the R-package `adjustedcranlogs`(**?**) finds that there are spikes in downloads due to automatic re-downloads and package updates, and it also provides a method to remove these download logs. Some of the methods are function extensions based on previous packages, and some of their proposed methods are new functions, but generally speaking, they are more tool like, to help users explore the characteristics of package download logs from as many directions as possible.

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

- A CRAN mirror is a website containing differently located servers, which aims to facilitate to quick and smooth access for people from different regions and countries. Each server is called a mirror (**?**).

- CRAN task view aims to provide some guidance regarding which packages on CRAN are relevant for tasks related to a certain topic (**?**).

2.1 Other work

`adjustedcranlogs`(**?**) : In this tool, 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 show the total daily down-

load counts 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 (**?**).

Chapter 4

Methods

In this project, we extracted R-package names from CRAN and obtained summary daily download logs of R-packages through the web Application Programming Interface (API) maintained by r-hub(**?**). We also collected release dates of R-packages through CRAN. In addition, we also tried to scrap the number of commits on master (main) branch in Github repositories for all of available R-packages. Then we conducted eight exploratory data analysis studies:

- we explored the daily downloads 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, yearly, on 1st April. from 2013 to 2021, to see how user preference change;
- we figured out the relationship between initial release dates and the total download counts over the most recent 6 month period. The reason for choosing the period of 6 months is : different R-packages are initially released in different time. If the comparison period is too long, it will be unfair to the R-packages released later. And if the period is too short, download data will be not enough to reflect some potential patterns or relationships;
- we compared two closely related packages : `fable` and `forecast`, to see the difference between their MA (moving average) trends;
- we studied how the number of commits on master (main) branches in Github repositories may influence the total download counts for the

most recent 6 month period;

- we explored the relationship between the number of updates for all of R-packages and their download counts;
- we analyzed the name length patterns for both CRAN task view packages and all of R-packages on CRAN;
- we explored how the alphabetical order of R-packages would related to the download counts and the statistic features (e.g.variance, median of the download counts).

Chapter 5

Results

5.1 Daily downloads of R-packages

Finding 1: There was unusual download activities 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 2021-06-12. The data was obtained from the `cranlogs` package(**?**), which includes a summary of the download logs via 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 **??**. The one happening in 2014 was on 2014-11-17, Monday, while the other one happening in 2018 was on 2018-10-21, Sunday.

When having a closer look into those two spikes, we firstly focused on the one on 2014-11-17. From Table **??**, we could see that the downloads of top downloaded R-packages on this day differs little, indicating this spike is not due to a certain package.

Table **??** shows the downloads from different countries on 2014-11-17. It is obvious that `Indonesia` obtains much more downloads than any others.

Furthermore, we also checked IP addresses, displayed in Table **??**. Downloads from `ip3758` is much higher than others. So, it seems that most of the downloads are owing to one certain IP for the unusual spike in 2014.

Next, we turned to the unusual spike in 2018. Table **??** shows the downloads from `tidyverse` is much higher than others, with nearly three orders of

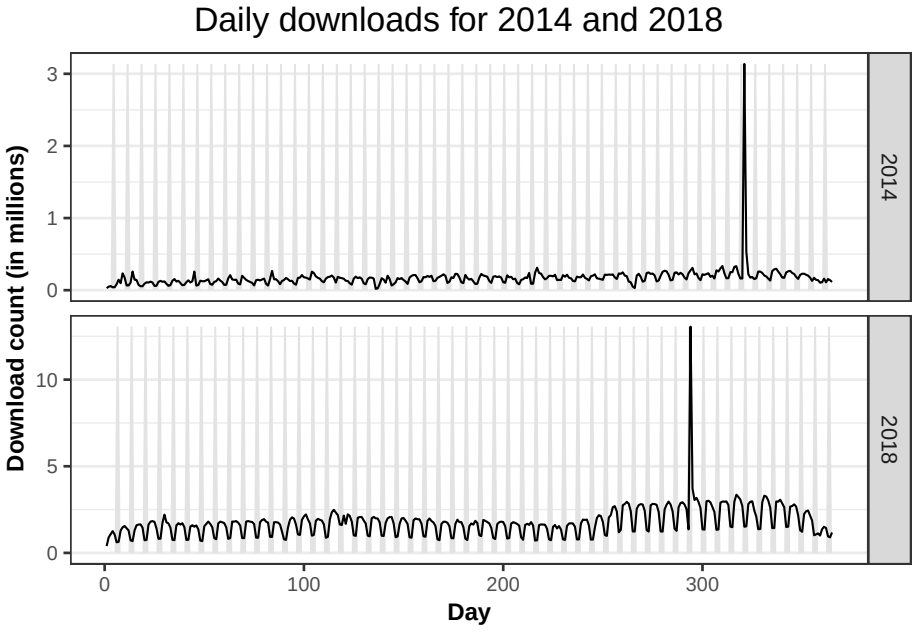

Figure 5.1: Unusual download spikes in 2014 and 2018.

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 addresses 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 2018-10-21

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 2018-10-21

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

magnitude.

As for country, from Table **??**, we could know that `US` occupies the most part of downloads on that day.

Finally, the most interesting finding is on IP address, displayed in Table **??**. Several consecutive IPs have highly distinguished downloads. It seems that they are probably from the same individual, or caused by a server test issue, in such a 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 quantities of downloads came from

Table 5.6: The IP addresses downloading from CRAN on 2018-10-21

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, it is guessed that they should come from the same individual, and it is very likely due to sever test issue, for it may be not necessary or reasonable for an individual to generate such a large amount downloads in one day.

Finding 2: There are increasing numbers of downloads over time, which can attests the growing number of R users.

Figure **??** shows the download trend of all R-packages on CRAN over a period pf time from 2012-10-01 to 2021-06-12, after fixing the unusual spikes mentioned above. There is an upward trend, with an increasing variance in download counts.

Finding 3: Weekends have a lower downloads than weekdays.

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

To be more specific, except for 2012 and 2013, the patterns of other years are very similar, with a strong weekly seasonality. To be more detailed, in 2012, the download logs showed an overall upward trend, which also reflected more and more users there after release of CRAN. 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 on weekends. Although the pattern of 2013 is more volatile, it still conforms to that. We suppose that is because CRAN was only open for a short period of time in 2013, so the

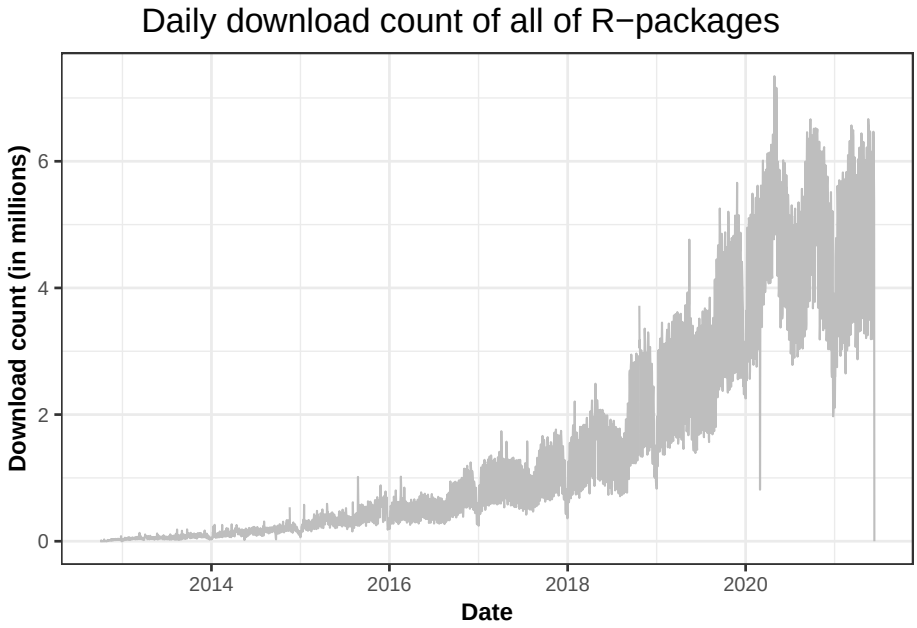

Figure 5.2: The download trend of all R-packages on CRAN from 2012-10-01 to 2021-06-12.

amount of download data is not adequate to show the weekly pattern very clearly. After 2016, the pattern of each year is quite consistent, for the total downloads have been increasing year by year. Back to weekly seasonality, people are more likely to download packages during weekdays, and rest on weekends. So, the trough of download curve always occurs on weekends. In addition, the lowest downloads across the year are always happening at the end of December or the beginning of January, probably due to the Christmas and New Year's holidays. Meanwhile, the downloads are on the rise from August to October, and from February to April, which covers the beginning of semesters for many universities around the world, a time when related students tend to download CRAN R-packages very often.

As there are many fluctuations in daily download pattern, which is due to calendar effect and test server issue of CRAN mirror, an STL decomposition model explained in **?**, was applied, to smooth the curve for all of the R-packages in Figure **??**.

Figure **??** shows the distributions and the median of the downloads between weekday and weekends, which differ from each other a lot. The violin plots of weekends are wider and shorter, while those of weekdays are thinner and higher, on the contrary. That is because the total downloads on weekends are less than those in weekdays. In 2012, the median and interquartile

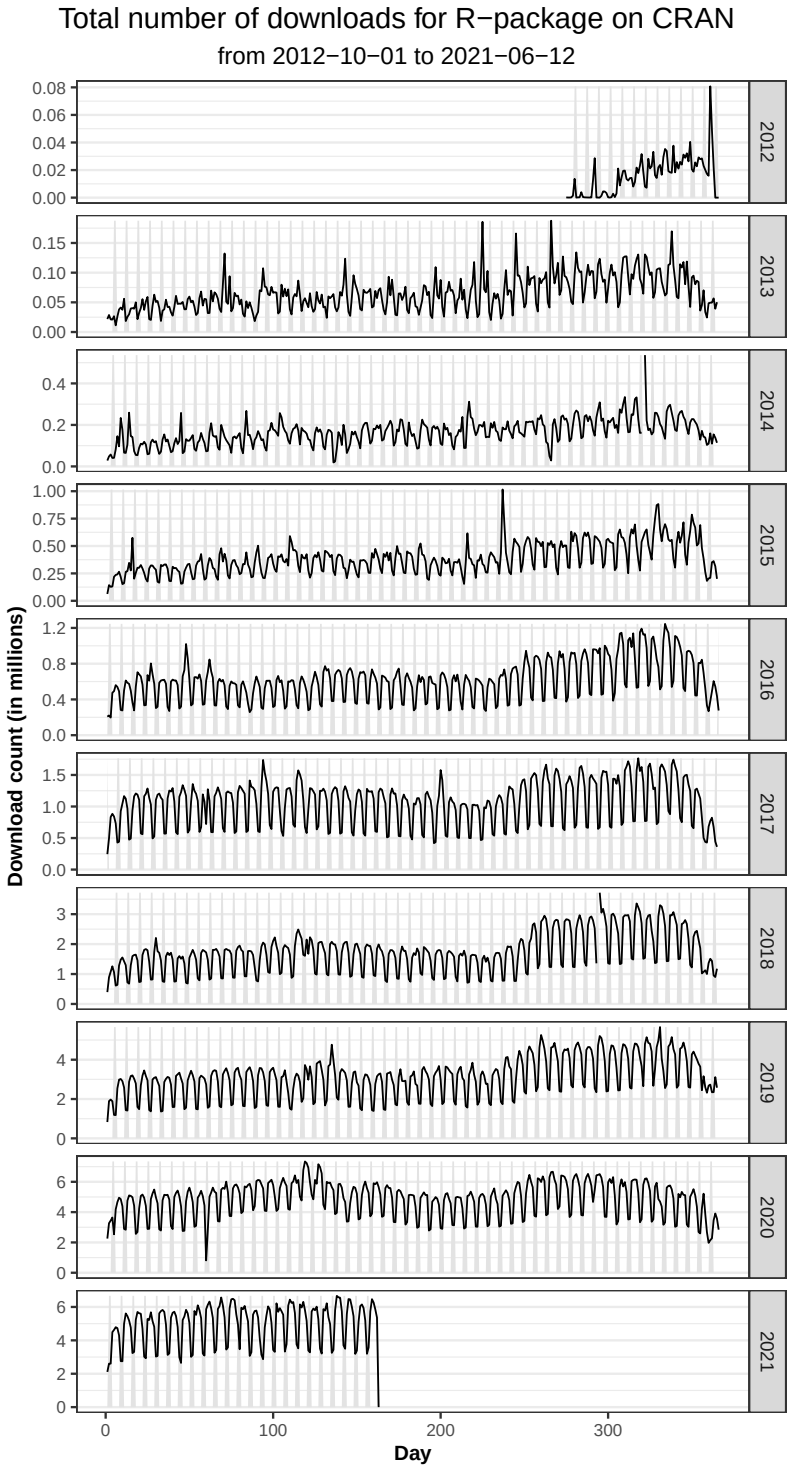

Figure 5.3: The total downloads for all of R-packages on CRAN would decrease on weekends and increase during weekdays.

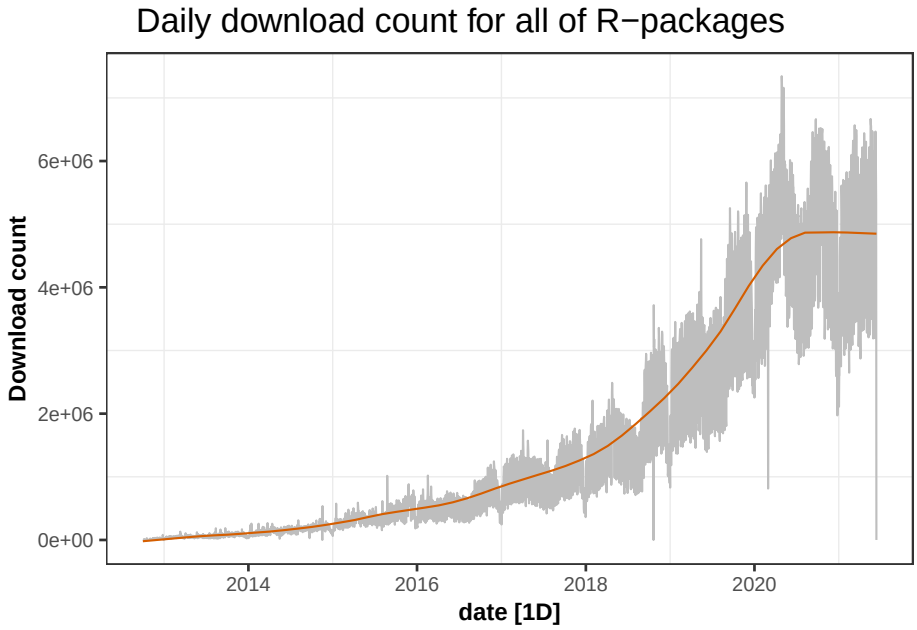

Figure 5.4: The total downloads of all R-packages on CRAN after smoothing.

range of download logs are not very distinguished between weekdays and weekends, for the data volume was not adequate at this time as mentioned before. But after 2013, the gap between the two has been becoming more and more obvious. The median downloads of working days are significantly higher than those of weekends, and the overall download volume is also significantly larger than that of weekends as well. Interestingly, the lower adjacent sometimes occurs on weekends, such as in year 2014, 2015, 2018, 2019 and 2021, while sometimes also in weekdays, such as in year 2012, 2013, 2016, 2017 and 2020.

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

From the previous analysis, we could see that the cumulative download counts of R-packages show 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 knowing from experience, we know that is hardly possible. So, here, we introduced Lorenz curve(**?**) to show the respective numbers of R-packages within different download levels (groups defined by quantiles of download

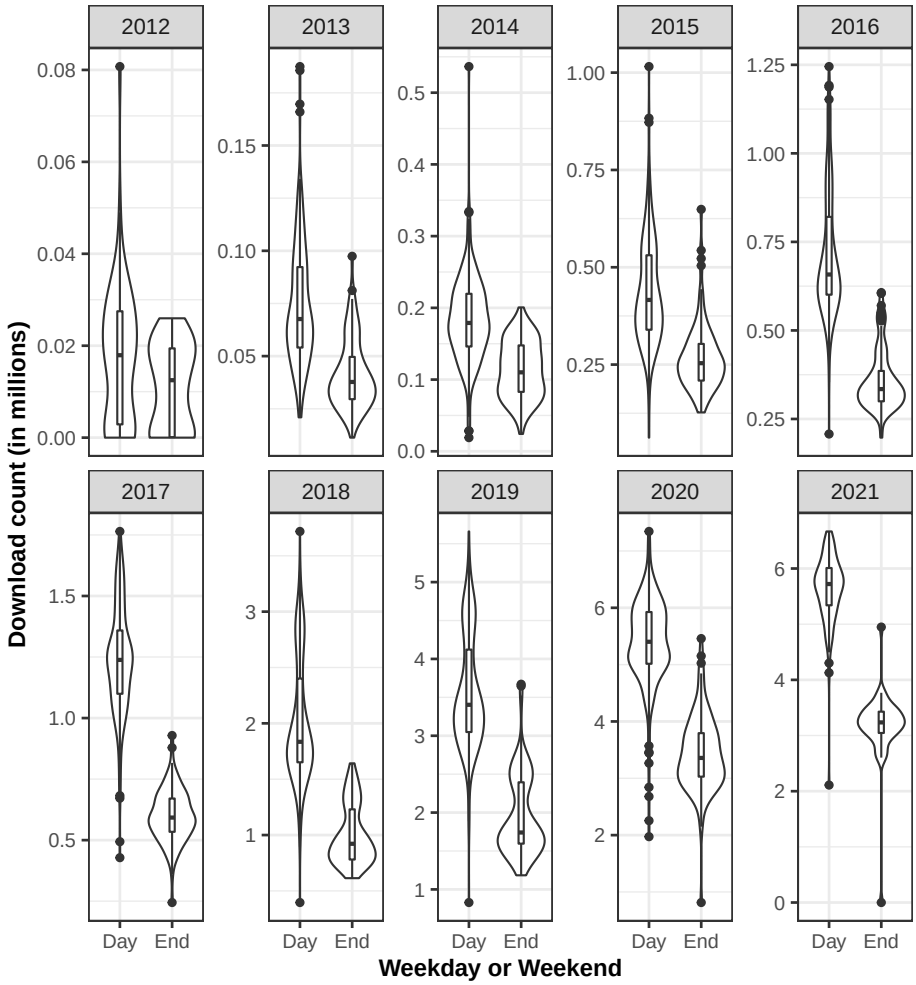

Figure 5.5: The violin plot for downloads of all of R-packages on CRAN, between weekday and weekends.

counts). In this way, we could figure out how many download counts contributed by different downloaded R-packages.

Figure **??** shows cumulative download counts 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 volumes across groups are quite 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.

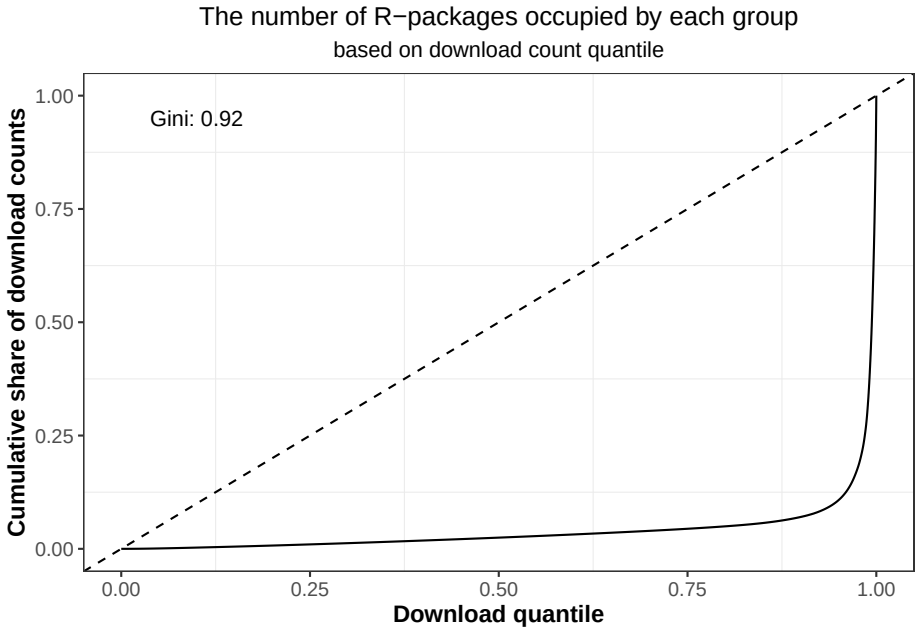

Figure 5.6: Percentiles of the download counts against cumulative download counts for R-packages at or below that percentile.

For example, if we extracted the first 10 packages of this group in Table **??**, we could find that there are many quite famous and frequently-used R-packages, such as `rlang` and `dplyr`.

\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 2021-06-12. The data was obtained from the `cranlogs` package (**?**) as well.

Finding 1: There is an upward trend in the number of downloads, also with an increasing variance.

Figure **??** shows the download trend of R on CRAN over the period of time, from 2012-10-01 to 2021-06-12. An upward trend shows, with an increasing variance in download counts, which is resulted from the growth of R users. As for significant spikes, they are probably quite similar to the situation of R-packages, which is due to repeated downloads, weekly calendar effect or server test issue.

Finding 2: The most used operation system for R users turns out to be Windows OS.

Figure **??** shows the comparison of operation systems for R users. The number of Windows-user ranks first, followed by MacOS and SRC. 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 only around 100 million active Mac users, up to now.(**?**)

Finding 3: The most popular version of R is 3.2.1

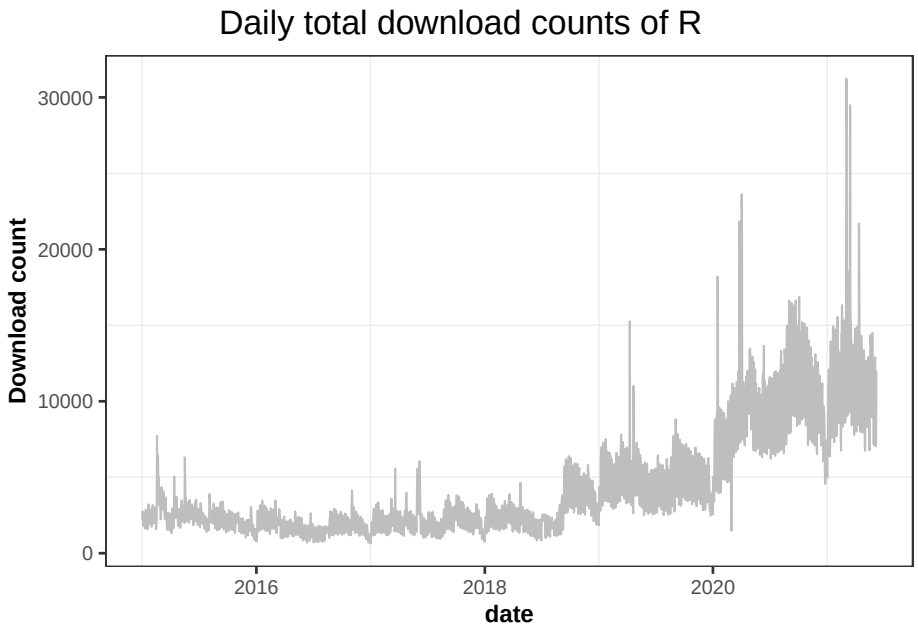

Figure 5.7: The download trend of R on CRAN, from 2012-10-01 to 2021-06-12.

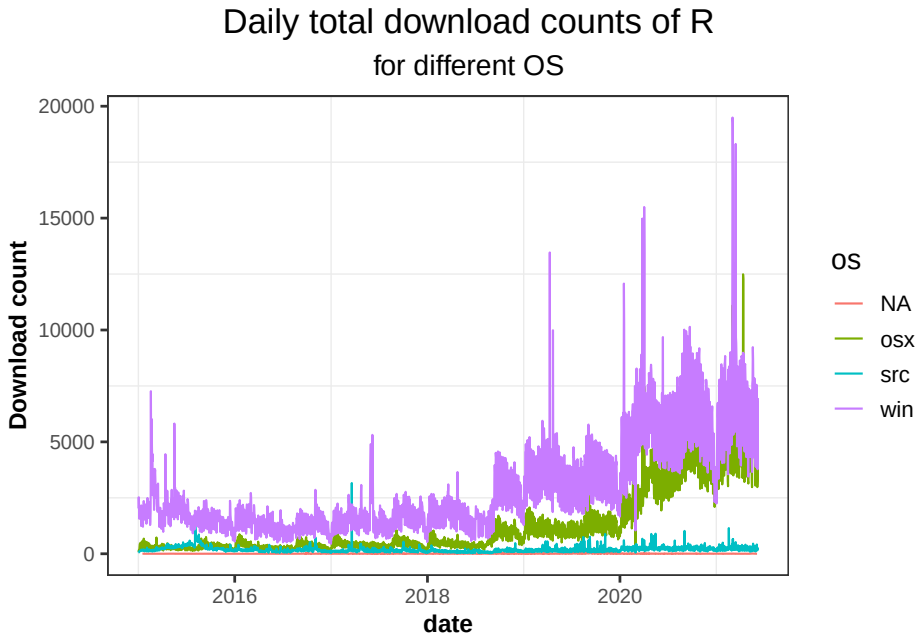

Figure 5.8: The comparison of operation system (OS) for R users, from 2012-10-01 to 2021-06-12.

Table 5.7: The numbers of downloads for 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

Table **??** shows the first 10 numbers of downloads for 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. Most users are likely to install the latest version of R to get the newest update information.

In conclusion, the daily download trends of R are quite similar to those of R-packages, and they are both significantly correlated to weekly calendar as well as unusual downloads from CRAN mirrors. 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 have been changing. The data were 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, some are developed by core personnel or personnel closely related to R, and some are created by prolific developers(**?**). And there are also parts of R-packages that are dependency packages of others, which means the increasing downloads of their “father” packages would also promote the downloads of them. For example, R-package `fable` is dependency package within R-package `fpp3`. When users install `fpp3`, `fable` would also be installed at the same time. Naturally, the numbers of downloads for those packages can probably be higher. On the one hand, the R-packages developed by those experienced developers may be more mature and completed. On the other hand, users are more inclined to use packages produced by famous authors, which can be regarded as a kind of “good

use" guarantee.

However, the existence of those packages can cause the results of user preferences biased. Therefore, we decided to explore the R-packages constructed by non-special creators, and 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 **?**)

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

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

Table **??** shows the R-packages that newly come up to the top 15 list each year, from which we can know how the user preferences have 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 R-package is `XLConnectJars`, related to JAVA dependency.
- For 2016, the first downloaded R-package 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 first-ranking `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.

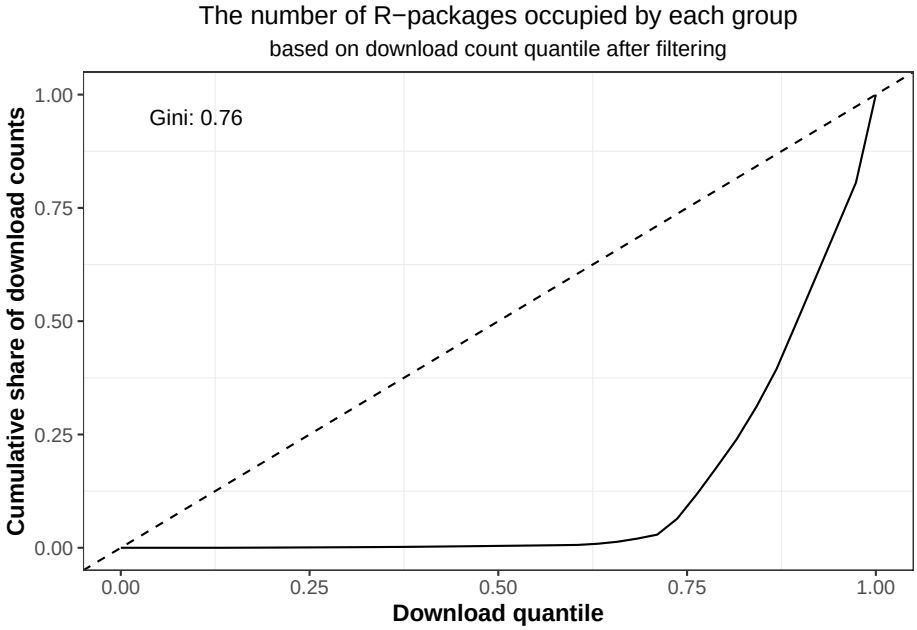

Figure 5.9: Percentiles of the download counts against cumulative download counts of R-packages after filtering four types of popular and highly-downloaded packages.

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	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-

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	ElemSt
rggobi	-	-	mixOmics	ElemStatLearn	reports
its	-	-	gWidgets	alr3	gWidg
-	-	-	-	maxent	d3heat
-	-	-	-	gWidgets	alr3

Finding 2: The topic of R-packages changes least during 2017 and 2019. The most popular application filed is on JAVA dependency.

Table **??** shows the R-packages that remain unchanged each year, compared to 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. They are used for providing JAVA dependency, data mining, linguistic research and providing API for building toolkit-independent, interactive GUIs.

Next, we changed the object to trending R-packages. Trending R-packages are ones downloaded at least 1000 times last week, which have increased significantly compared to the average weekly downloads in the previous 24 weeks(**?**). That is to say, they are packages with significantly high download volume in a recent short time. Through their topics, we can know what application areas of R-packages that people are concerned about recently.

Finding 3: The most popular topic of trending R-packages.

Table **??** shows the top 10 downloaded topics for trending R-packages. The most popular topic is Multivariate, followed by Cluster and Phylogenetics. This is easy to understand : the numbers of users from different fields are different, so the downloads of R-packages in different topics also varies. It seems that Multivariate is quite heated recently.

Apart from the topics of R-packages, what other factors can probably be

Table 5.10: Ranking topics of trending R-packages

topic	download times
Multivariate	4
Cluster	3
Phylogenetics	3
TimeSeries	3
WebTechnologies	3
Distributions	2
ExperimentalDesign	2
SocialSciences	2
ClinicalTrials	1
Finance	1

linked to download volumes? With this question in mind, we then explored the relationship between the total number of download counts for all of CRAN R-packages and the earliest release date in the past year. We also studied the relationship between the total download volume and the number of updates, the number of commits on master (main) branch in GitHub repositories, the lengths and the alphabetical order of names, over the most recent 6 month period, in the following sections.

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

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

In our common cognition, it may be assumed that the earlier an R-package is released, the more people can get to know it, and thus the more downloads it can have. However, R-packages related to different topics cannot be directly compared, because download counts of R-packages in one topic can be higher than that in another. Therefore, in order to test this conjecture as clearly as possible, we selected three domain R-packages through CRAN task view(**?**), calculated their respective downloads in the previous one year, and extracted their earliest release

dates for comparison. Those three topics are :

- 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(**?**).

- 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(**?**).

- Econometrics R-packages

The last topic is related to econometrics. 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(**?**).

Figure **??** displays the scatterplot of the past year's download counts and the earliest release dates, for `Time Series Analysis`, `Econometrics` and `Bayesian` R-packages. It can be seen that, generally, as the earliest release dates get later and later, the numbers of download logs become lower and lower. For `Time Series Analysis` R-packages, they are mainly released between 2012 and 2019. For `Bayesian` R-packages, most of the R-packages are born from 2007 to 2012. And most `Econometrics` are centered between 2013 and 2016.

In conclusion, it is not surprising to find that the earlier the R-package is released, the more downloads it could have, which is reflected in all of three topics above. That is probably because the R-packages released earlier can be better-known. When they are released early, there may be a relatively small number of R-packages in the same topic, under non-serious competition. As a result, the R-packages coming later can easily be covered up, since people may 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 a senior user of an R-package, they may recommend that R-package to their students when teaching, especially when they obtain a satisfying user experience.

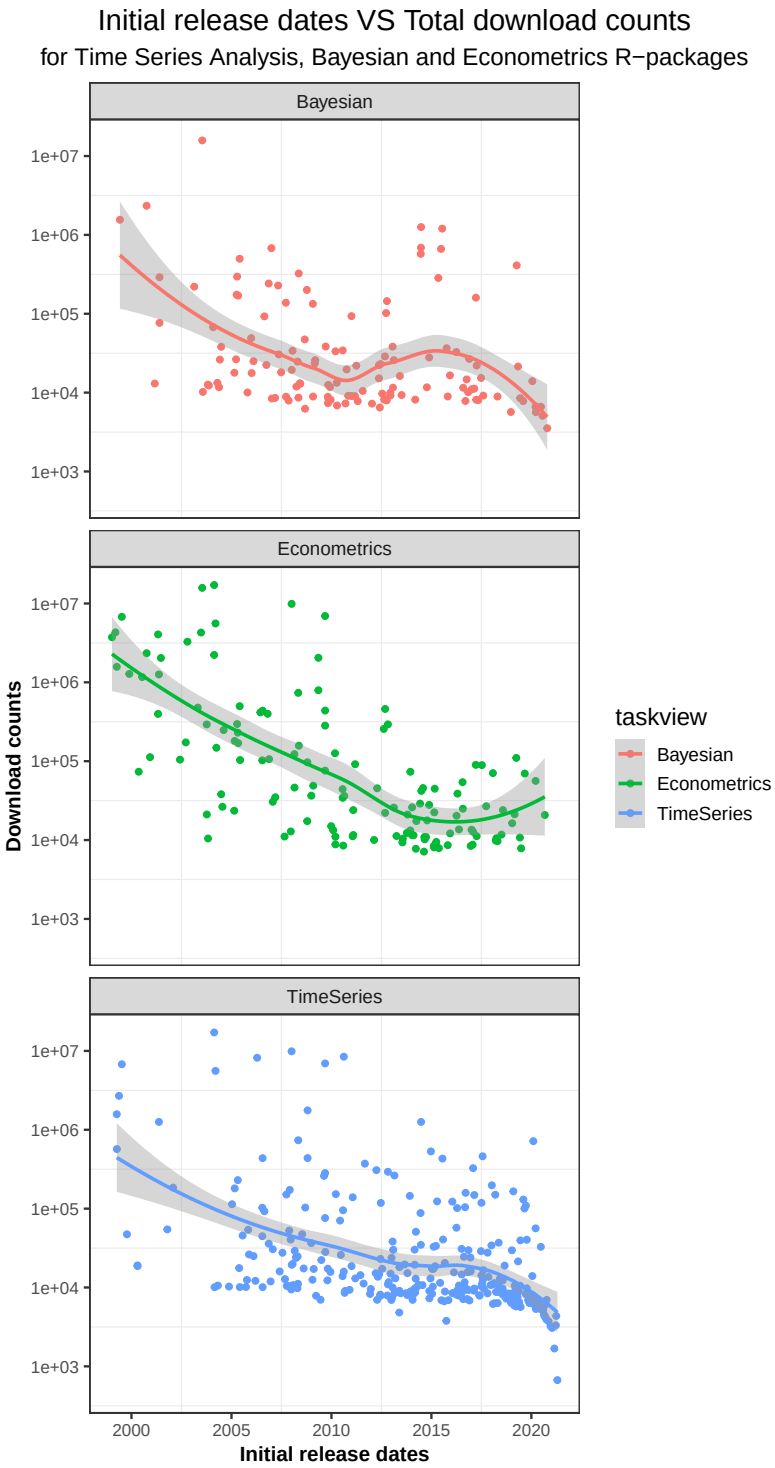

Figure 5.10: The download counts decrease with the initial release dates.

5.5 Compare moving average trends for fable and forecast

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

As stated in the previous section, the earlier an R-package is released, the easier it can be to get a relatively higher total downloads. 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 of factors, except the initial release date, so that we can compare the growth and changes of those two more clearly.

Figure **??** and Figure **??** show the daily download counts changing over the most recent 6 month period, with strong weekly seasonality. That means the downloads tend to be higher within week days and thus lower on weekends, which is consistent with trend for all of R-packages that was analyzed before.

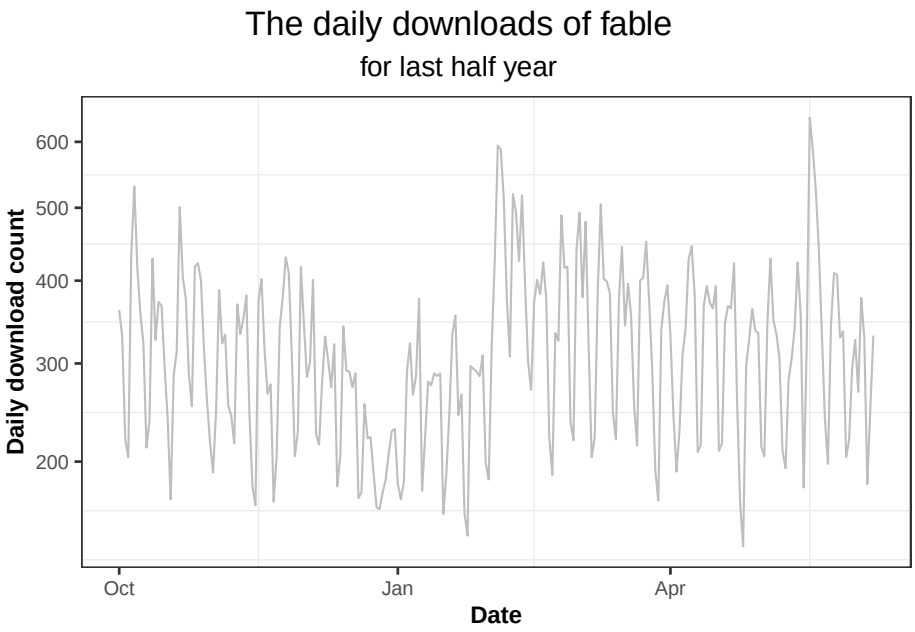

Figure 5.11: The daily downloads of R-package "fable"

Therefore, in order to estimate the trend-cycle and reduce the weekly

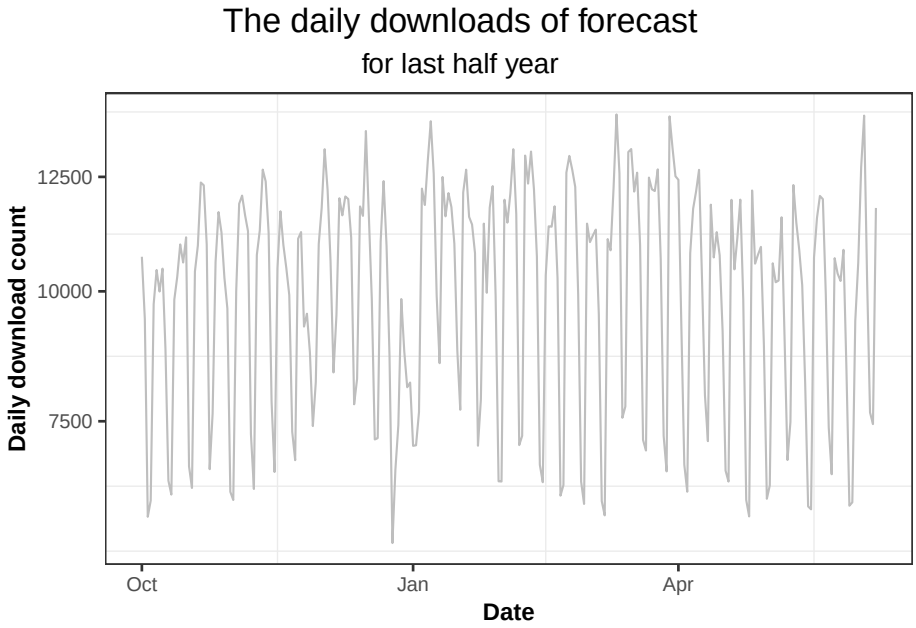

Figure 5.12: The daily downloads of R-package “forecast”.

seasonality, to see the changes more clearly, the Moving Average (MA) was introduced(**?**).

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 an equal weighed 7 moving average. 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 **??** shows the moving average (MA) of `fable` and `forecast` respectively. They have quite different moving average patterns, when `forecast`’s download volume is much higher than that of `fable`, except for the time around New Year’s Eve, with a significant drop in `forecast`. 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, on 2021-1-29. Soon after that day, `fable`’s downloads peaked, which was due to the increase brought by

update day. At the same time, `forecast` drops, then only gets a gentle increase.

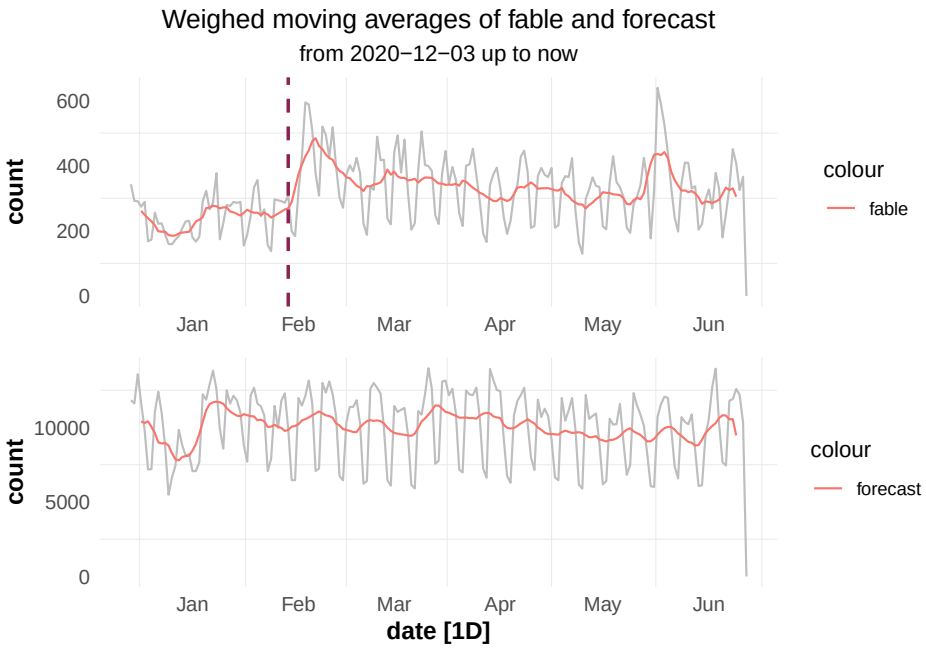

Figure 5.13: The moving averages of R-package “forecast” and “fable”.

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. It coincides with the conclusion of the previous section : the earlier released R-package tends to obtain more cumulative downloads. In addition, we can also see that the growths of both of two sometimes increase at the same time, while the growth of one can correspond to the decline of the other, sometimes. As for as we concerned, the shared growth may be due to their similarity. Users may download both at the same to compare, like us. And the trade-off may also be due to their similar functions, one can be replaced by the other. Finally, earlier release does not guarantee a faster growth. For example, `fable` has a more dramatic growth than `forecast` from 2021-01-21 to 2021-02-06.

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

In this section, we compared total downloads over the most recent 6 month period, with the number of commits on master branch in Github

repositories. For all of CRAN R-packages, there were only 6185 R-packages that have Github repositories, and after cleaning up, only 5769 remained.

Finding: In our initial assumption, more commits on master branch of Github repositories are likely to bring R-packages more download counts. On the one hand, the number of commits can indicate that developers are constantly supplementing and updating their R-packages, which may lead to more downloads. On the other hand, the number of commits also can reflect the attention developers attach to their R-packages, to some extent. Usually, for those who have more commits, their developers may advertise more or try other ways, to expand the popularity and improve the download counts.

The way we initially planned to apply was extracting the commits through accessing the Github REST API(**?**). To achieve that, we first scraped the Github URLs from 'description' pages for all of R-packages from CRAN, then cleaned up the multiple URLs and other redundant symbols or characters. The URL 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"*. 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 contents of URLs through Github API, for all of CRAN R-packages.

Based on that, Table **??** shows the first 5 R-packages, with their Github URLs and the `user_repo` parts.

However, the Github API has a rate limits, allowing for up to 60 requests per hour for unauthenticated requests, which can be extended to 5000 per hour after authentication(**?**). But even after getting authentication, our rate limit didn't get promoted. So, we switched to scrape commits with Python spider, by setting random user agent, to avoid the API limit. Since this method is quite time consuming (around 5 hours each time for all available R-packages), we saved the output as a local .txt file and load it later in R studio, to speed up the code execution.

\begin{table}

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

Table 5.11: R-packages with commits on Github and total downloads over the most recent 6 months

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 **??** shows the first 5 R-packages of all, along with their total downloads over the most recent 6 month period, and the numbers of commits on master branch in Github repositories.

Figure **??** shows the scatterplot, along with a smoothing line. In general, more commits can link to more downloads.

Another method can be conducted by taking the ultra-low-downloaded R-packages into consideration, whose download counts only rank last 1% of all. Another purpose for this way is to show our initial idea on scraping commits through Github API with R (as the sample size is less than 60 R-packages, on this condition).

Table **??** shows the last 10 quantiles of download counts, it can be seen that the last 1% download count is around 375.12. As the download counts of those packages are extremely low, we could assume that many factors can have little effect on their downloads, and the only two differences among them are the numbers of commits on Github master branch and the total download counts.

Based above, we could select those ultra-low-downloaded R-packages and extract commit counts from their Github repositories. In Figure **??**, it could be observed that when the number of commits increases from 0 to 100, the download counts first increase and then decrease. After that, the download volume keeps rising, with a small jump at the end. We consider that might result from the too small-sized sample, and the observation causing the

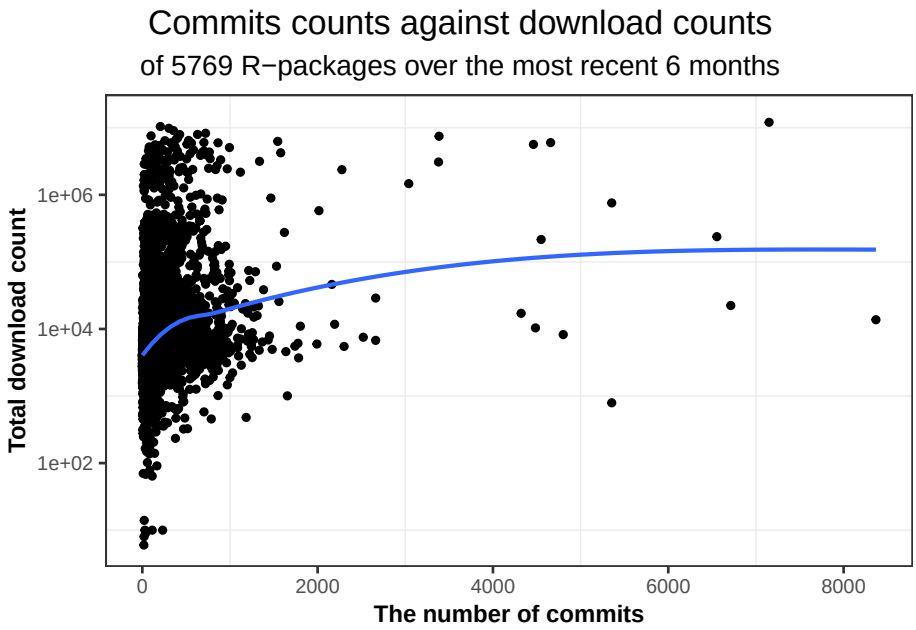

Figure 5.14: The commits on master branch in Github repositories against the total download counts over the most recent 6 month period.

decline can be an outlier. If expanding the observation horizon, an increase in this curve can be expected. Therefore, it can be concluded that even for those last 1% downloaded R-packages, the phenomenon - “the more commits can link to more downloads” also exists.

5.7 Compare download counts with the numbers of updates

In this section, we looked into the relationship between the numbers of updates and total download counts over the most recent 6 month period, for all of CRAN R-packages.

Finding 1: The download counts tend to rise with the numbers of updates. This is probably because there will be a significant increase in downloads when packages get updated. During that time, not only old users often download the latest version, but also it is easier to attract new users, for the increase of downloads within a short period of time can bring packages up to the trending list[@r-hub].

Table 5.12: Quantile of total download counts for all of R-packages on CRAN

	download_count
0%	0.00
1%	375.12
2%	721.56
3%	1075.34
4%	1340.24
5%	1562.70
6%	1674.00
7%	1804.46
8%	1903.24
9%	1968.02

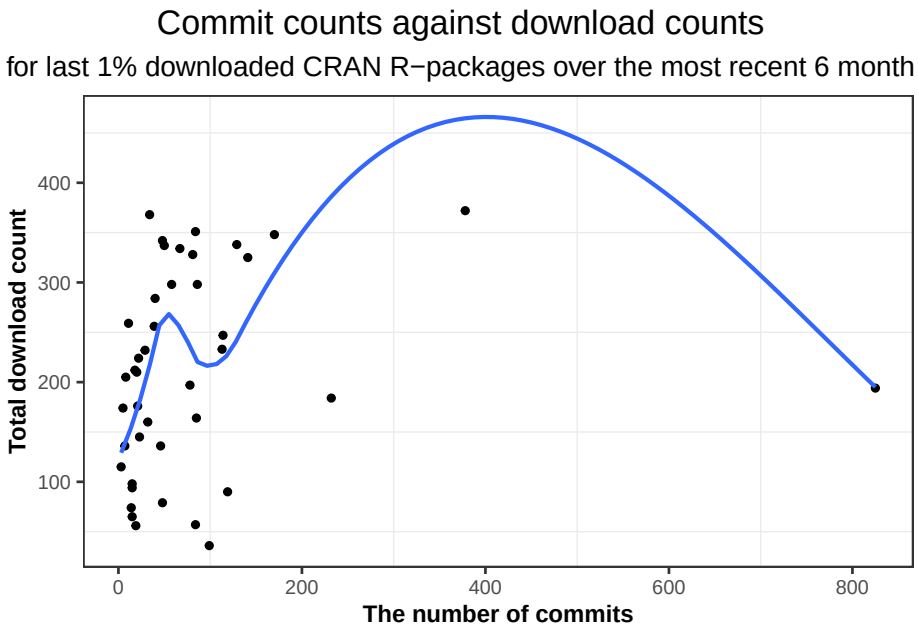

Figure 5.15: The commits on master branch of Github repositories, against the total download counts over the most recent 6 month period, for last 1% downloaded R-packages on CRAN.

Table 5.13: Percentage of CRAN R-packages whose updates are less than average

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

Figure **??** shows that the numbers of downloads increase with the update times. And most R-packages are updated no more than 30 times.

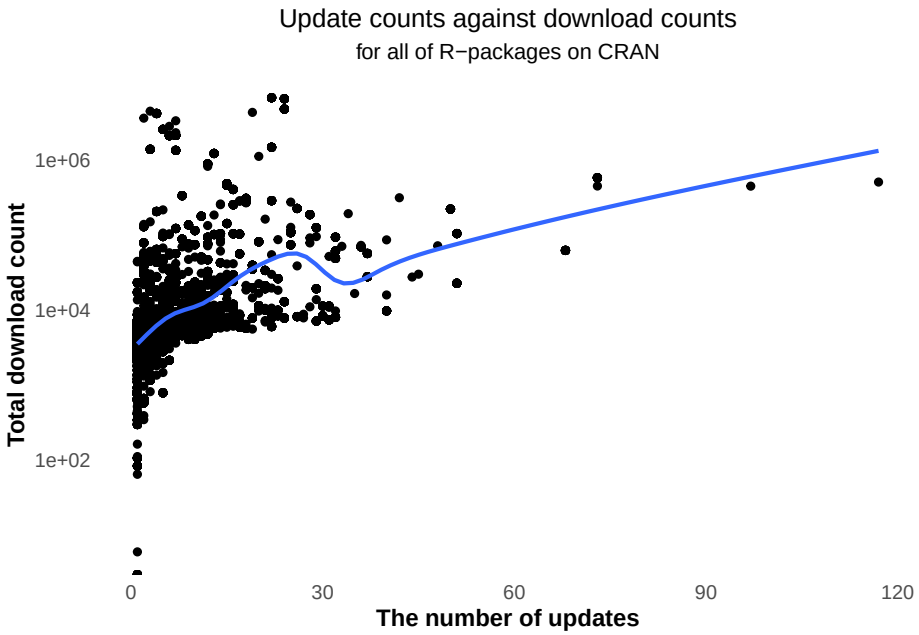

Figure 5.16: The download counts increase with the numbers of updates.

Finding 2: Over half of the CRAN R-packages are not likely to update very frequently.

By checking Table **??**, 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 CRAN R-packages do not tend to update very frequently.

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

It can be seen from Figure **??** that most R-packages' latest publish dates are after 2015, which indicates that many R-packages are likely to update with the time, probably in order to keep their activity.

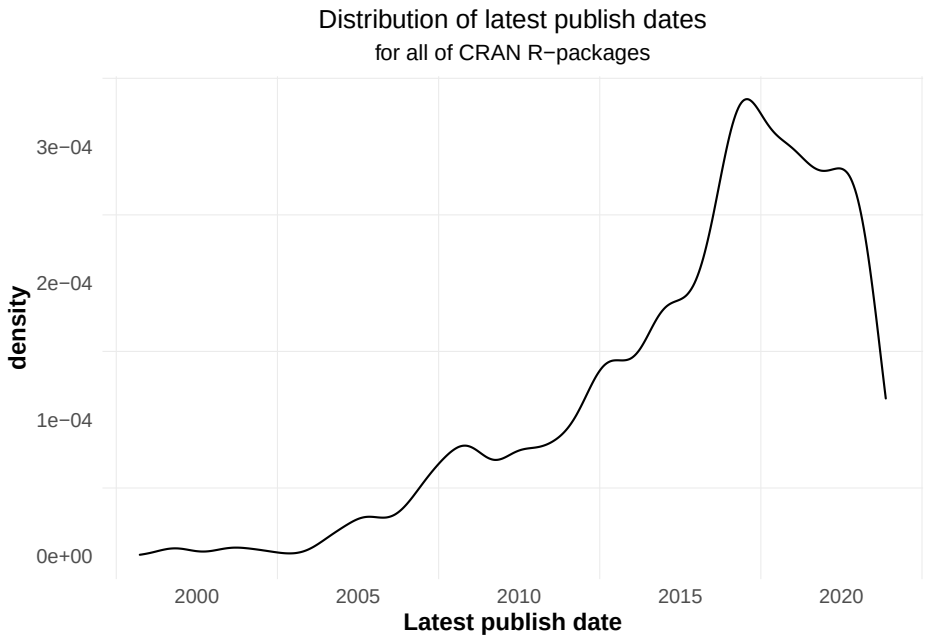

Figure 5.17: The latest update dates of CRAN R-packages are almost the recent dates.

Finding 4: Most of the CRAN R-packages are likely to update with a relatively long time interval.

Figure **??** shows that with the increase of update intervals, the numbers of downloads first increase and then decrease slightly. Most of the time intervals are between 45 and 450 days, which shows that their update frequency is not very high.

In conclusion, it's not that the more updates R-packages have, the popular they can be. In actual, most of the CRAN R-packages whose updates are lower than average occupy the majority. Therefore, the numbers of updates are probably not that important for R-packages. The really important thing can be keeping updated along the time.

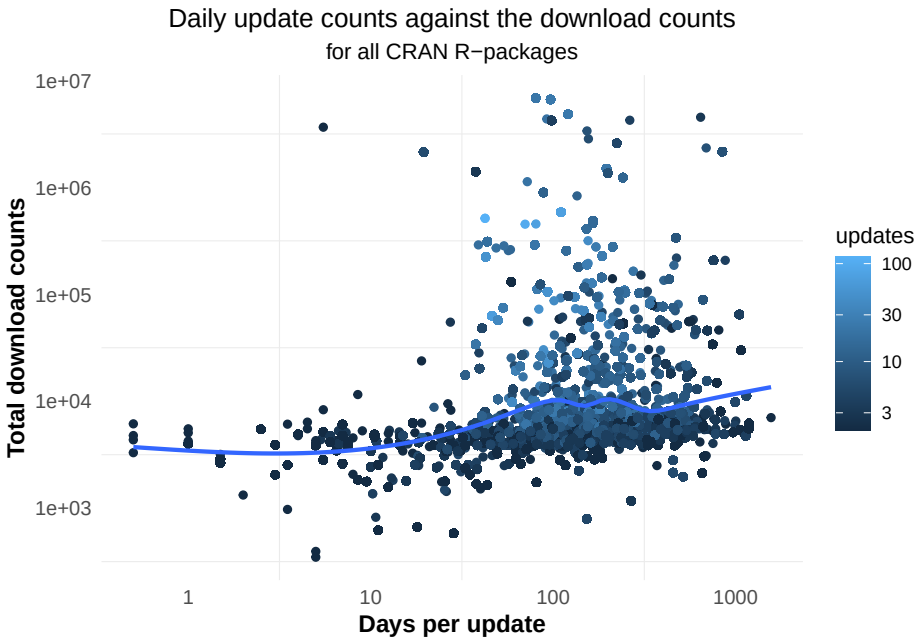

Figure 5.18: Most of the time intervals for updates are between 45 and 450 days.

5.8 Compare the package name lengths with download counts

Here, we still focused on all of R-packages on CRAN, and made comparison between their name lengths and total downloads over the most recent 6 month period.

Finding 1: The name lengths of R-packages have no significant correlation with total downloads, over the most recent 6 month period.

We could see from Figure **??** that, the influence on download volume resulted from name length is not obvious. But we could still observe that the name lengths of most of R-packages are centered before 10 characters long. The names , with more than 6,000,000 downloads, are between 5 and 9 characters long. The most downloaded one is R-package rlang whose name length is 5.

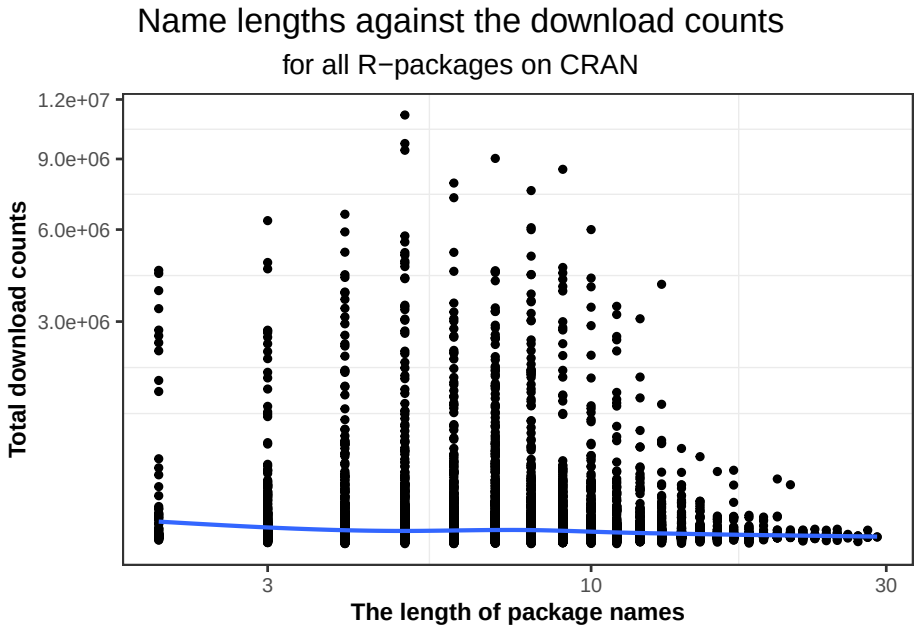

Figure 5.19: The names of R-packages with more than 6,000,000 downloads are between 5 and 9 characters long.

Table 5.14: Percentage of packages whose name lengths are below average

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

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

Table **??** shows that the average name length of all the R-packages is 7.8461146. And over half of the CRAN R-packages are more likely with name lengths below average. And R-packages with shorter names can be easier to get relatively higher downloads. That may because shorter named packages are easier for users to remember.

After finding that there is no obvious relationship between the name lengths and the download volume, a new question came up : Can the name lengths of the R-packages be linked to the time of initial release date?

Finding 4: For task view R-packages, the name lengths increase with the initial release dates, especially for Bayesian packages.

We may have this kind of experience in life : for example, detective novels, the later they are released, the less names can be chosen, because many names have been occupied by the books published earlier, with the same theme. Therefore, those later published books often have to lengthen their names to distinguish themselves from the existing books. Coincidentally, we guessed the naming of R-packages from the same topic would also be correlated to the initial release time. So, we looked back to the CRAN task view R-packages(**?**), for conducting comparison among R-packages from the same topic. Figure **??** shows the name lengths of CRAN task view R-packages against the initial release dates. It is obvious that the name lengths of task view R-package tend to increase with the initial release dates, especially for `Bayesian` R-packages.

Finding 5: For all of 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 R-packages within the same topic, we also had a view on the annual change in the average name length, for all of R-packages on CRAN.

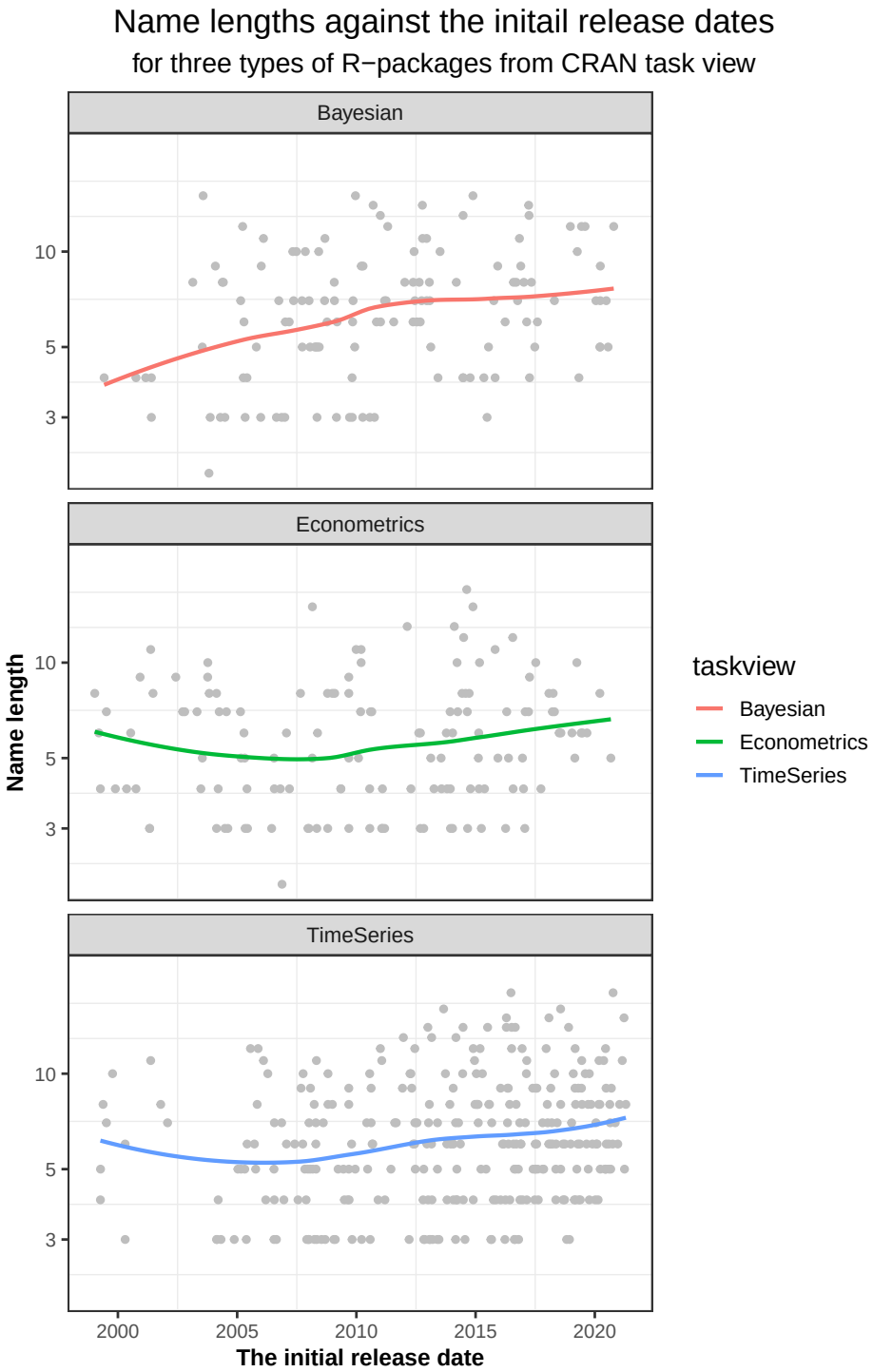

Figure 5.20: The name lengths of task view R-packages slightly increase with the initial release dates.

Figure **??** shows the average name length for all of R-packages on CRAN, released in each year. It is obvious that the name lengths of those R-packages generally increase year by year.

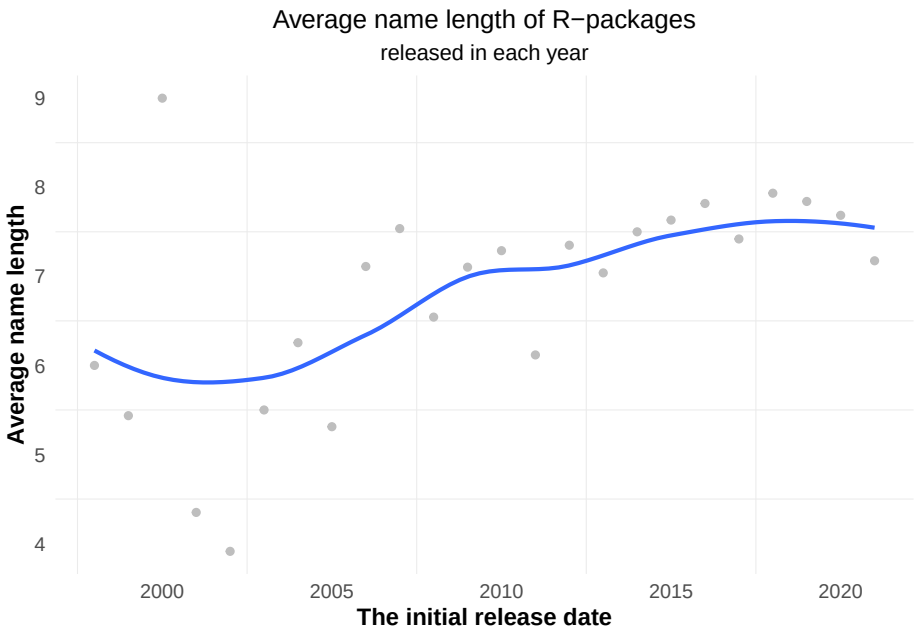

Figure 5.21: The average name length for all of R-packages on CRAN released in each year, tends to rise along the time.

5.9 Compare download counts with alphabetical order of names

We might also have such experiences in our life : when we go shopping in the supermarket, the products placed in front of the shelves can be noticed by us easier. From the perspective of R-packages, in addition to the lengths of names, we also wondered whether alphabetical order can link to the download volume. R-packages with earlier alphabetical order will be placed at the first part of package list on CRAN(**?**). To answer this question, we grouped the R-packages by 26-letter order, calculated the average downloads for each group, and made comparisons.

Finding 1: For all of CRAN R-packages, the average downloads of different alphabetical groups are slightly increasing with alphabetical order, while the total downloads tend to decrease a little, on the contrary.

From Figure **??**, we could see that the average downloads of different alphabetical groups are slightly increasing with alphabetical order, while the total downloads tend to decrease, instead. That is because the later-ordered groups contain fewer R-packages. Developers may prefer to name their packages with top alphabetical order, which might be easier for users to notice.

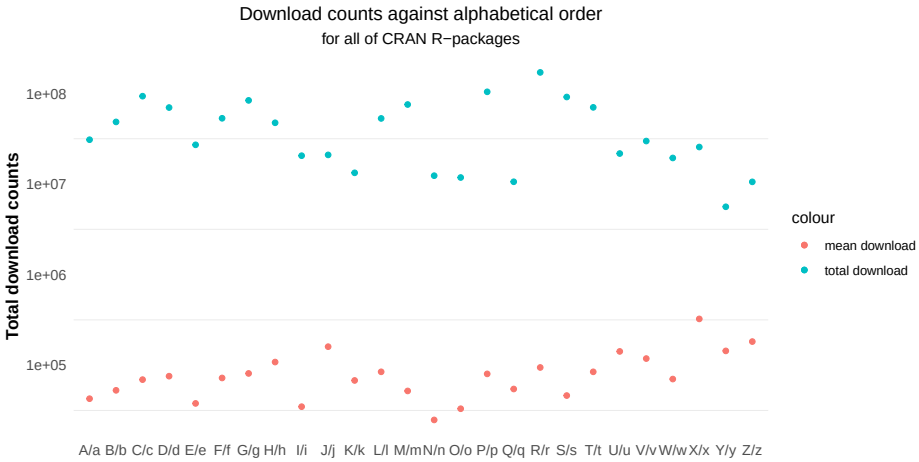

Figure 5.22: The average total download counts of each group is little linked to the alphabetical order of name, for all of R-packages.

Finding 2: For all of R-packages on CRAN, alphabetical groups with higher total downloads tend to have greater variance, owing to more outliers.

Then, we took a look at how the variance varies across groups. Figure **??** 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. The variances across groups difference little, which means that for each group, 50% of the download counts are relatively concentrated. The real difference is the highest and lowest downloads per group. In general, the larger the total numbers of downloads (which also means more packages in this group), the more outliers can be included, such as group "F/f", "L/l" and "R/r".

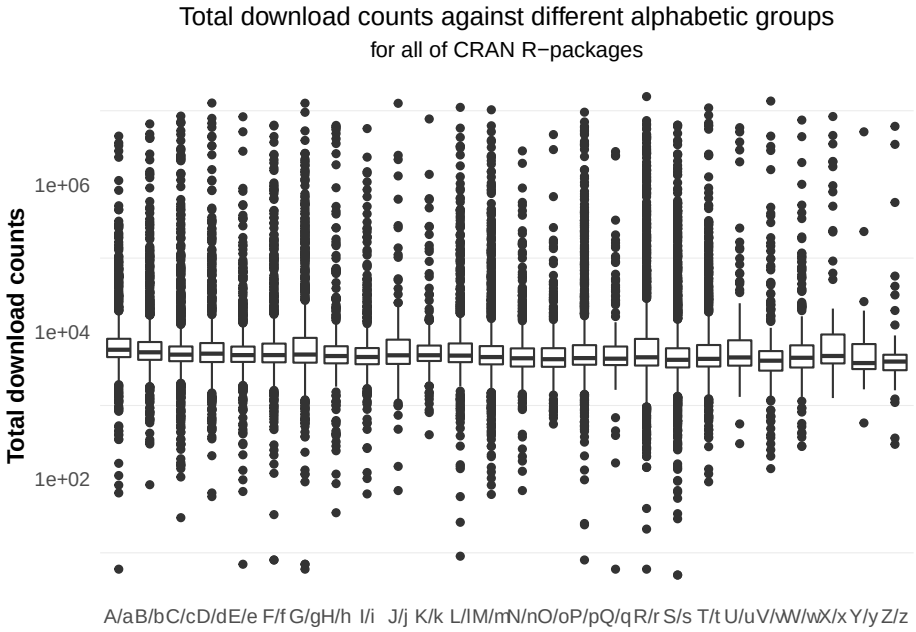

Figure 5.23: The R-packages with name starting with "j" has the largest variation.

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

Therefore, we could approximately draw a conclusion : In general, the R-packages with top alphabetical order can be easier to get relatively higher download volume, with non-significant gaps. At the same time, the higher the numbers of downloads, the greater the variance can appear in this group.

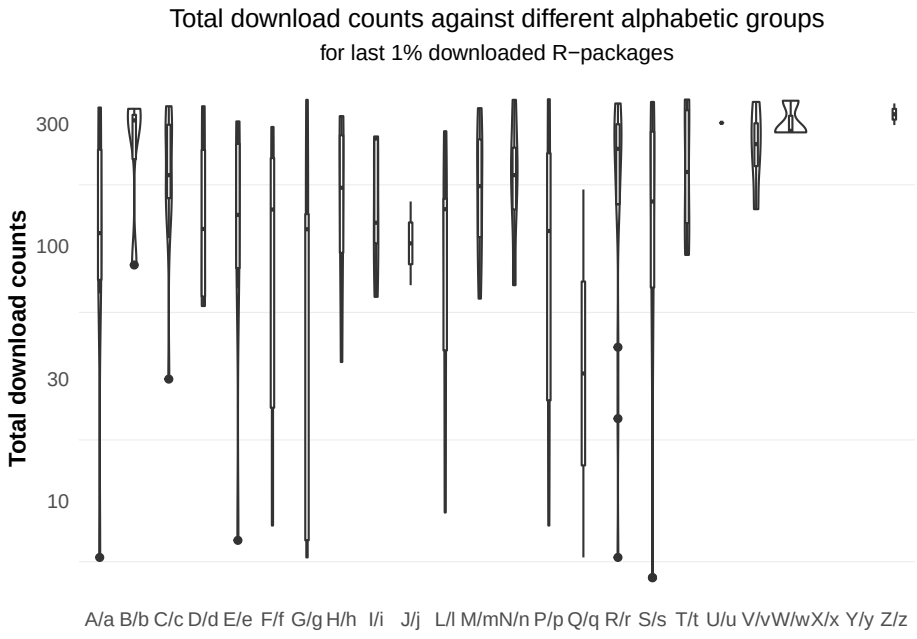

Figure 5.24: The last 1% downloaded R-packages with name starting with “U/u” has only one observation.

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(**?**) and also used daily download data in CRAN for a time period from 2012-10-01 to 2021-06-12, to explore the daily download pattern for all of R-packages each year. In that case, we found that the cumulative numbers of downloads increase over time, with an also increasing variance, which indicates that some R-packages with larger downloads grow rapidly. In addition, there is also a strong weekly seasonality in the daily download pattern. The download counts tend to peak through weekdays and drop on weekend. What's more, through Lorenz curve, we also found that most of the cumulative downloads came from the top 10% downloaded R-packages, which means the distribution of downloads is quite unequal. Part of the reason is that those top 10% downloaded R-packages contain quite a lot popular and frequently used R-packages, such as `tidyverse` and `rlang` that would probably obtain 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 **?**)

However, the existence of those R-packages may make it difficult to reflect the popularity of other R-packages, so we excluded them, for the analysis of user preferences. And we found that the topics of newly added R-packages on 1st Oct of each year are from quite different application

areas, while the R-packages remaining most stably popular during 2017 and 2019 is related to JAVA dependency. Definitely, JAVA always rank top three among programming languages according to TIOBE Index(**?**).

As for R itself, its download pattern is quite similar to that of total R-packages on CRAN. 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, we extracted the release dates for all of CRAN R-packages and task view R-packages, to compare the total download counts (past year for task view R-packages and the most recent 6 month for all) among R-packages with different release dates or with different numbers of updates. And we found that for R-packages from the same topic, earlier release date can usually related to more download counts, while R-packages with more update times would not always have higher downloads. 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 numbers of commits in Github repositories, for all of available CRAN R-packages, through Github API by R, to check whether more commits can result in more downloads. But there came a tricky problem on the rate limit of Github API. As documented in **?**, 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 promoted. In that case, we switched to make this done with Python by setting random user agent. 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 R method. Therefore, we could expect that, generally, if an R-package has more commits on master (main) branch in Github repository, it can probably obtain more download counts.

The last two parts are about analysis for R-package names. We compared the average downloads among R-packages with different name lengths 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. Alphabetical order played little roles in promoting the download volume.

Chapter 7

Conclusion

In conclusion, we hold the belief that the download counts can be a relatively simple and reliable measurement for popularity of CRAN R-packages, and there are many factors that could be linked to it, such as the popularity of the creator, the application fields, the release dates, whether to keep updated, and the lengths of their names. Anyway, we could generally believed that a relatively popular R-package should have earlier release date, shorter name and maybe more commits in Github repository if they have. There is also another point that could not be ignored : it is better to keep updated.