We can try to quantify the development work of authors by looking at the  
number of lines deleted and added in the git repo of the package before,  
during, and after review. By number of lines we mean number of lines in  
any file in the repo so it can be lines of actual code and lines of docs  
(in the README for instance). To do so, we link information from the  
package git repos (information about sizes and dates of commits) and  
information from the onboarding issue threads (start and end dates of  
the review process, as measured by open/close dates).

library("ggplot2")

library("magrittr")

commits <- readr::read\_csv("output/gitsum\_reports.csv")

issues <- readr::read\_csv("data/clean\_data.csv")

issues <- dplyr::group\_by(issues, package) %>%

dplyr::summarise(opened = min(created\_at),

closed = max(closed\_at))

commits <- dplyr::left\_join(commits, issues, by = "package")

custom\_transf <- function(x){

pof <- log(x)

pof[x == 0] <- 0

pof

}

custom\_exp <- function(x){

pof <- exp(x)

pof[x < 0] <- - exp(-x)

pof

}

plot\_commits <- function(package\_name, commits){

message(package\_name)

example <- dplyr::filter(commits, package == package\_name, !is\_merge)

p <- ggplot(example) +

geom\_segment(aes(x = datetime, xend = datetime,

y = 0, yend = custom\_transf(total\_insertions)),

col = "salmon", size = 1) +

geom\_segment(aes(x = datetime, xend = datetime,

y = 0, yend = - custom\_transf(total\_deletions)),

col = "blue", size = 1) +

ylab("lines") +

annotate("segment",

x = min(example$datetime),

xend = max(example$datetime),

y = 0, yend = 0,

col = "grey50",

size = 0.2) +

scale\_y\_continuous(breaks= seq(min(-custom\_transf(example$total\_deletions)),

to = max(custom\_transf(example$total\_insertions)),

by = 5),

labels= round(custom\_exp(seq(min(-custom\_transf(example$total\_deletions)),

to = max(custom\_transf(example$total\_insertions)),

by = 5))))+

annotate("rect",

xmin = example$opened[1], xmax = example$closed[1],

ymin = - custom\_transf(max(example$total\_deletions)),

ymax = custom\_transf(max(example$total\_insertions)),

fill = "grey80", alpha = 0.6,

col = "grey10",

size = 0.2) +

ggtitle(package\_name) +

hrbrthemes::theme\_ipsum()

print(p)

}

Let’s have a look at a few packages. The x-axis represents time, and the  
grey box the onboarding period (from issue opening i.e. submission to  
issue closing i.e. approval). The salmon lines above the y-axis are the  
number of lines *added* in one commit while the blue lines below the  
y-axis are the number of lines *deleted* in one commit.

plot\_commits("opencage", commits = commits)

commits plot of the opencage
package

plot\_commits("charlatan", commits = commits)

commits plot of the charlatan
package

When looking at all onboarded repos, we don’t see a general pattern to  
commit histories. Commits have diffent sizes, and the activity frequency  
is highly variable. While we see the updates to packages that occur  
during review, there is just as often change again after onboarding, as  
packages continue to improve and be maintained. This highlights an  
important aspect of our peer review, which is that acceptance is not the  
end of the process. While we aim for the process to ensure high quality  
at acceptance, packages are not static moving forward. Not all  
deficiencies are detected by peer reviewers, and further improvements  
and maintenance will continue. This is why many of our standards are  
about maintainability, and our process is in part an approach to onboard  
authors into a community of dedicated and supportive package  
maintainers.

We do try to onboard mature packages, i.e. that are not *drafts*.  
Furthermore, the absence of general patterns in the previous figures  
could be due to different age at submission. How old are packages at  
submission?

age <- commits %>%

dplyr::group\_by(package) %>%

dplyr::summarise(age = difftime(min(opened), min(datetime), units = "weeks")) %>%

dplyr::filter(age > 0)

library(hrbrthemes)

ggplot(age) +

geom\_dotplot(aes(age), binwidth = 5) +

xlab("Age (weeks)") +

scale\_y\_continuous(NULL, breaks = NULL) +

hrbrthemes::theme\_ipsum(base\_size = 16,

axis\_title\_size = 16)

age of packages at
submission

On this dotplot, the x-axis represents age at submission and each dot is  
a package falling in that age bin. We say *apparent* age because the  
first commit might happen a long time after the package was created,  
although often one makes an initial commit not long after having started  
to work. We needed to filter positive age because in one case the GitHub  
repo of the onboarded package was apparently deleted and re-created  
without history after approval. Such things happen, thankfully the  
package wasn’t lost, just its history!

Many packages are submitted while still very young which might indicate  
rapid development. It might also mean some authors viewed onboarding as  
a part of development, i.e. authors knew they intended to submit quite  
rapidly. For instance, Maëlle developed opencage in just a few days,  
then submitted it to get it ready for wider use by the community.

How much effort is put in by reviewers in this process? Reviewer time  
and effort is one of our most precious resources . We  
measure this effort by asking reviewers to self-report the hours spent  
reviewing. We do so to understand the amount of effort we are asking  
reviewers for, so we can let new reviewers know what to expect, and so  
hopefully in the future we can measure the success of efforts to  
automate some reviewer tasks.

# airtable data

# this is our private database of who's reviewed what

# and of reviewers' areas of expertise

airtable <- airtabler::airtable("appZIB8hgtvjoV99D", "Reviews")

airtable <- airtable$Reviews$select\_all()

We have self-reported reviewing times for 136 of 184 reviews.

ggplot(airtable) +

geom\_boxplot(aes(y = review\_hours, x = "")) +

hrbrthemes::theme\_ipsum(base\_size = 16,

axis\_title\_size = 16) +

theme(axis.title.x=element\_blank()) +

ylab("Reviewing time (hours)")

hours spent
reviewing

One potential question is whether reviewer time is affected by the size  
of the package reviewed as measured by for instance number of exports  
(classes and functions).

get\_namespace <- function(package\_name){

message(package\_name)

local\_path\_archive <- paste0(getwd(), "/repos\_at\_submission/", package\_name)

if(length(fs::dir\_ls(local\_path\_archive)) != 0){

ns <- devtools::parse\_ns\_file(local\_path\_archive)

tibble::tibble(package = package\_name,

exports = length(ns$exports) +

length(ns$exportClasses))

}else{

return(NULL)

}

}

packages <- fs::dir\_ls("repos\_at\_submission")

packages <- stringr::str\_replace\_all(packages, "repos\_at\_submission\\/", "")

purrr::map\_df(packages, get\_namespace) %>%

readr::write\_csv("output/namespace.csv")

Interestingly, we find no relationship between the reviewing time and  
number of exports:

namespace\_ro <- readr::read\_csv("output/namespace.csv")

namespace\_ro <- dplyr::left\_join(namespace\_ro, airtable, by = "package")

ggplot(namespace\_ro) +

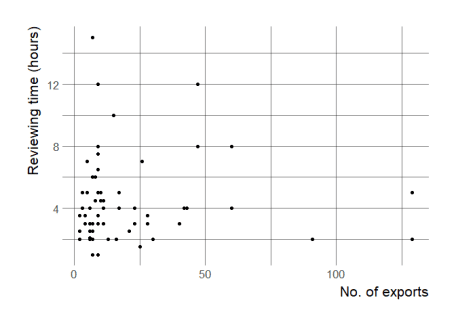
geom\_point(aes(exports, review\_hours)) +

hrbrthemes::theme\_ipsum(base\_size = 16,

axis\_title\_size = 16) +

ylab("Reviewing time (hours)") +

xlab("No. of exports")



There are a few potential explanations to this that might be fruitful to  
explore. For instance, does this mean that there’s only so much time,  
and so larger packages get less scrutiny per line of code or per export?  
Or does review time just depend more on the reviewer than the package?

**Work done by editors**

Editors manage the review process, performing initial package checks,  
identifying and contacting reviewers, and then moderating and cajoling  
the process forward. Our best measure for editor effort is the number of  
packages handled by an editor in a given time frame, which we can track  
from the assignments on GitHub issues since the start of onboarding.  
Looking at this over time, we can see how editor workloads changed in  
response to growing number of assignments and how we have attempted to  
manage this by expanding our editorial board:

library(tidyverse)

library(gh)

library(lubridate)

issues <- gh("/repos/ropensci/onboarding/issues?state=all&labels=package", .limit=1000)

edits = map\_df(issues,

~data\_frame(url = .$html\_url,

editor = .$assignee$login %||% NA\_character\_,

opened = as.Date(.$created\_at))) %>%

filter(!is.na(editor)) %>%

mutate(quarter = paste(year(opened), quarter(opened), sep="Q"),

half = paste(year(opened), if\_else(quarter(opened) <= 2, 1, 2), sep="H"),

year = year(opened))

edits %>%

group\_by(editor, half) %>%

summarize(n\_assigned = n()) %>%

{ full\_join(., crossing(editor = unique(.$editor), #can't get expand() to work.

half = unique(.$half))) } %>%

mutate(n\_assigned = coalesce(n\_assigned, 0L)) %>%

ggplot(aes(x=half, y=n\_assigned, fill=editor)) +

geom\_col(position="dodge") +

geom\_hline(yintercept = c(3, 6)) +

xlab("Half / Year") + ylab("No. Issues Handled")+

hrbrthemes::theme\_ipsum() +

theme(legend.position = "bottom") +

viridis::scale\_fill\_viridis(discrete = TRUE)

number of assignments per editor per half a
year

As an side-note, the same data allows to visualize the increase in the  
number of submissions.

edits %>%

group\_by(half) %>%

summarize(n\_submissions = n()) %>%

ggplot(aes(x=half, y=n\_submissions)) +

geom\_col(position="dodge") +

xlab("Half / Year") + ylab("No. Submissions")+

hrbrthemes::theme\_ipsum()

number of submissions per half a
year