Supplementary

Siruo Wang

Code Section 1

```
library(dplyr) # Reference [1]
library(gh) # Reference [2]
library(lubridate) # Reference [3]

# start from 2008-01-01
date_start <- ymd("2008-01-01")        ## start date
day_inc  <- 14                        ## increment days by 14 at a time
dates <- c()
i <- 1

# create dates from 2008-01-01 to now in 14 days period
while(date_start < Sys.Date() - (day_inc+1)) {
  dates[[i]] <- c(rep(date_start,2) %m+% c(days(-1),days(day_inc+1)))
  date_start <- date_start + days(day_inc + 1)
  i <- i + 1
}

# read token from githubtoken.txt, which is saved in .gitignore and not avail
able to the public
token <- readLines("githubtoken.txt")
# loop through pages to get repository names based on 14 days period starting
from 2008-01-01 to now
# rest for 60 seconds after pulling data from 10 pages with 100 repository na
mes each to get pass API excess error
dates_repos <- lapply(c(1:length(dates)), function(dates_num) {
  gh_date <- paste("created:", paste(dates[[dates_num]], collapse=".."), sep=
"")
  repos <- c()
  for (page_num in 1:10){

    repo_name <- paste0("GET /search/repositories?q=getting+and+cleaning+data
+",
                        gh_date, "&per_page=100")
    x <- try(gh(repo_name, page = page_num, .token = token))

    if ("try-error" %in% class(x) == FALSE) {

      repos <- c(sapply(x[[3]], "[[", "full_name"),repos)
    }

  }
```

```
  print(dates[[dates_num]])
  Sys.sleep(60)
  return(repos)
})

save()
# save all repository names into all_repos.rda
save(dates_repos, file = "all_repos.rda")
save(dates, file = "dates.rda")
```

Code Section 2

```
library(gh) # Reference [2]
load("all_repos.rda") # load: dates_repos
repolength <- lapply(c(1:length(dates_repos)), function(x) {length(dates_repo
s[[x]])})
do.call(sum,repolength) # how many repos in total

repos <- unlist(dates_repos)
token <- readLines("githubtoken.txt")

# create an empty rfile list before scrapping data
rfile_list <- list()
for (i in 1:length(repos)){
  rfile_list[[i]] <- NA
}

# count how many of them are NAs after scrapping data from Github
numof_unfinished <- 0

for (i in 1:length(repos)){
  string <- paste0("GET /search/code?q=repo:", repos[i],"+extension:r")
  res <- try(gh::gh(string, .token=token), silent = TRUE)

  if ("try-error"  %in% class(res) == FALSE) {
    # loop_path to get all .R files
    path <- try(res[[3]][[1]]$path, silent = TRUE)

    if ("try-error" %in% class(path) == FALSE) {

      code.url <- file.path("https://raw.githubusercontent.com",repos[[i]], "
master", path)
      # fix space problem in file path to github
      code.url <- gsub(" ","%20",code.url)

      tryCatch({code <- readLines(code.url, warn = FALSE)
                rfile_list[[i]] <- code},
               error = function(x) {message(x)},
               warning = function(x) {message(x)})
```

```
      Sys.sleep(5)
    }
    else {
      print(paste0("no r file found in the repo: ",repos[i]))
      numof_unfinished <- numof_unfinished +1}
  }

}


print(paste0("number of unfinished: ",numof_unfinished))

# save all R scripts from the Getting and Cleaning Data project to rfile_list
.rda
save(rfile_list, file = "rfile_list.rda")
```

Code Section 3

```
packages_list <- c("ggplot2", "gridExtra", "lattice")
packages_need <- packages_list[!(packages_list %in% installed.packages()[,"Pa
ckage"])]
if(length(packages_need) > 0) install.packages(packages_need)



library(ggplot2)   # Reference [4]
library(gridExtra) # Reference [5]
library(lattice) # Refence [6]

load("dates.rda") # load: dates (saved in Code Section 1)
load("all_repos.rda") # load: dates_repos (saved in Code Section 1)
repolength <- lapply(c(1:length(dates_repos)), function(x) {length(dates_repo
s[[x]])})
do.call(sum,repolength) # how many repos in total

## [1] 20824

repolength_unlist <- unlist(repolength)

timeline <- c()
for (i in c(1: (length(dates)))){
  current <- as.character(dates[[i]][1])
  timeline <- c(timeline, current)
}

start_idx <- which(repolength>2)[1]
time_lab <- timeline[start_idx:length(repolength_unlist)]
time_periods <- c(1:length(start_idx:length(repolength_unlist)))
repositories_time_df <- data.frame(repositories = repolength_unlist[start_idx
:length(repolength_unlist)],  time = time_periods)
```

```
p1_repositories <- ggplot(repositories_time_df,aes(x = time, y = repositories
)) +
        geom_point() +
        geom_line() +
        scale_x_continuous(breaks = time_periods, labels = time_lab) +
        theme(axis.text.x = element_text(angle = 270, hjust = 1),plot.title =
element_text(hjust = .5))
png("p1_repositories.png",width = 1080, height = 1080, units = "px")
grid.arrange(p1_repositories)
dev.off()

## quartz_off_screen
##                 2
```

Code Section 4

```
load("rfile_list.rda")
rfile_length <- unlist(lapply(rfile_list,length))

drop_out_rate <- sum(rfile_length==1) / length(rfile_length)
completion_rate <- 1 - drop_out_rate
completion_rate

## [1] 0.9407415
```

Code Section 5

```
load("rfile_list.rda")
# get pure text by omitting comment lines and space lines and count for pure
text length for each student
pure_text <- lapply(rfile_list, function(x) {x[!grepl("^#", x) & !grepl("\\ ^
#", x) & x != "" & x != " "]})
pure_text_length <- unlist(lapply(pure_text,length))
# set up a filter for only getting students code when the pure text length is
between 5 to 300
rfile_length_filter <- rfile_length[rfile_length<300 & rfile_length > 5]
pure_text_filter <- lapply(rfile_list[rfile_length<300 & rfile_length > 5], f
unction(x) {x[!grepl("^#", x) & !grepl("\\ ^#", x) & x != "" & x != " "]})
pure_text_length_filter <- unlist(lapply(pure_text_filter,length))

# create a data frame with full code length and pure code length for each stu
dent
rfile_puretext_filter_dataframe <- data.frame(pure_code = pure_text_length_fi
lter, full_code = rfile_length_filter)
# plot the data frame using ggplot2 and save it
p1 <- ggplot(data = rfile_puretext_filter_dataframe, aes(x = full_code, y = p
ure_code))+
    geom_point(alpha = 0.8, color = "steelblue") +
    ylab("pure code length") +
    xlab("full code length")
```

```
png("code_length.png",width = 1080, height = 1080, units = "px")
grid.arrange(p1)
dev.off()

## quartz_off_screen
##                 2
```

Code Section 6

```
# check for library usage

load("rfile_list.rda")
# only check for library usage in the pure text (ignoring the appearence of l
ibrary in the comments)
pure_text <- lapply(rfile_list, function(x) {x[!grepl("^#", x) & !grepl("\\ ^
#", x) & x != "" & x != " "]})
pure_text <- pure_text[!is.na(pure_text)]
pure_text_length <- unlist(lapply(pure_text,length))

# if the pure text length is less than 1, we definitely have an empty R scrip
t. We filter those out
pure_text <- pure_text[pure_text_length > 1]
pure_text_length <- unlist(lapply(pure_text,length))
rfile_text <- unlist(pure_text)
# we grep lines that have "library(" in it.
library_usage <- rfile_text[grep("library\\(", rfile_text)]

# combine libraries by merging different formats of the same library
library_usage <- sapply(library_usage, function(x) {
  gsub("\'|\t|\"| |>|prepare_|load_|\\{|\\}","",x)})

# handle multiple suppressmessage() cases
# run code in a for loop since some students have suppressWarnings(suppressMe
ssage(library(dplyr)))
for (i in 1:2){
  toMatch <- c("suppressMessages", "suppressPackageStartupMessages", "suppres
sWarnings","ifelse","capture.output","try")
  suppressmessage_idx <- grep(paste(toMatch,collapse="|"),library_usage)

  library_usage[suppressmessage_idx] <- sapply(library_usage[suppressmessage_
idx], function(x) sub("suppressMessages\\(|suppressPackageStartupMessages\\(|
suppressWarnings\\(|ifelse\\(|capture.output\\(|try\\(","",x))
  library_usage[suppressmessage_idx] <- sapply(library_usage[suppressmessage_
idx], function(x) sub(")","",x))
}

# handle "," in library() and "#" after library()
library_usage <- sapply(library_usage, function(x) sub('\\s*,.*',')', x))
library_usage <- sapply(library_usage, function(x) sub('\\s*#.*','', x))
```

```r
# handle cases when giving a value to library() using "<-" and some other spe
cial cases
library_usage <- sapply(library_usage, function(x) sub('.*<-|Enterfilecontent
shere|if\\(.*\\)|Installing','', x))
# handle cases where using ";" between library usages
library_subset <- library_usage[grep(";",library_usage)]
# split the lines using ";" and only getting the library() part out
split_list <- strsplit(library_subset,";")
library_subset_list <- c()
for (i in c(1:length(split_list))){
  for ( j in c(1:length(split_list[[i]]))){
    library_subset_list <- c(library_subset_list,split_list[[i]][j])
  }
}
library_subset_list <- library_subset_list[grep("library\\(",library_subset_l
ist)]
# handle some other special cases
library_usage <- c(library_usage[-grep(";",library_usage)],library_subset_lis
t)
library_usage <- library_usage[-which(library_usage == "" | library_usage ==
"}")]
library_usage <- library_usage[-which(library_usage == "dplyr_check==FALSE)"
| library_usage == "tidyr_check==FALSE)" | library_usage == "<tdid=LC1class=b
lob-codeblob-code-innerjs-file-linelibrary(<spanclass=pl-smireshape2</span)</
td")]
# handle two cases with special charactor <U+201D>
library_usage[library_usage == "library(data.table<U+201D>)"] <- "library(dat
a.table)"
library_usage[library_usage == "library(dplyr)<U+201D>)"] <- "library(dplyr)"

library_usage <- sapply(library_usage, function(x) gsub("library\\(plyr", "li
brary\\(plyr\\)", x))
library_usage <- sapply(library_usage, function(x) gsub("library\\(plyr)\\)",
"library\\(plyr\\)", x))
library_usage <- sapply(library_usage, function(x) gsub("library\\(dplyr", "l
ibrary\\(dplyr\\)", x))
library_usage <- sapply(library_usage, function(x) gsub("library\\(dplyr)\\)"
, "library\\(dplyr\\)", x))
library_usage <- sapply(library_usage, function(x) gsub("library\\(dplyr\\)m\
\)", "library\\(dplyr\\)", x))


# find unique library usages and count the frequency of unique library usage
unique_library <- unique(library_usage)
length(unique_library)

## [1] 156

unique_library_counts <- sapply(unique_library, function(x) {sum(library_usag
e %in% x)})
```

```
# sort the unique library usage by their counts and get the top 25 used libra
ries
top_used_libraries <- sort(unique_library_counts, decreasing = TRUE)[1:30]
# make plots for the top 25 used libraries
library_dataframe <- data.frame(counts = top_used_libraries[1:10],
                                library = names(top_used_libraries[1:10]))
p2 <- ggplot(library_dataframe, aes(x = reorder(library,counts), y = counts))
+
  geom_bar(stat="identity", fill="steelblue") +
  geom_text(aes(y = counts,label = counts), position = position_dodge(width=0
.9),hjust=0,color = "black",size = 2.5) +
  coord_flip() +
  xlab("library") +
  ylab("library frequency counts")
  ggtitle(paste0("A")) +
  theme(axis.text.x = element_text(angle = 0, hjust = 1),plot.title = element
_text(hjust = 0))

## NULL
```

Code Section 7

```
# library usage over time

unique_library_filter <- top_used_libraries
# use log length of pure code as response, and the usage of each top 25 used
library (record as 0 or 1) as predictors
# build a linear regression model (lm) and look at their coefficients
libusage_matrix <- matrix(data = 0, nrow = length(pure_text),ncol = length(un
ique_library_filter))
for (i in c(1:length(pure_text))){
  lib <- unique(library_usage[names(library_usage) %in% pure_text[[i]]])
  if (!length(lib) == 0){
    idx <- which(names(unique_library_filter) %in% lib)
    libusage_matrix[i,idx] <- 1
  }
}

libusage_df <- as.data.frame(libusage_matrix)
colnames(libusage_df) <- names(unique_library_filter)
libusage_df <- cbind(pure_text_length,libusage_df)
libusage_matrix_filter <- libusage_matrix[,1:10]
libusage_time <- c()
start <- 1
breaks <- 978
for (i in c(1:20)){
  end <- start + breaks -1
  if (end > length(pure_text)){
    breaks_sum <- colSums(libusage_matrix_filter[start:length(pure_text),])
    libusage_time <- rbind(libusage_time, breaks_sum)
```

```
    break}
  breaks_sum <- colSums(libusage_matrix_filter[start:end,])
  libusage_time <- rbind(libusage_time, breaks_sum)
  start <- end+1
}
colnames(libusage_time) <- names(unique_library_filter[1:10])
libusage_time_vector <- as.vector(libusage_time)
libusage_time_df <- data.frame(libusage = libusage_time_vector, lib = rep(nam
es(unique_library_filter[1:10]), each = 20), time = rep(c(1:20),10))

library_usage_overtime <- ggplot(libusage_time_df,aes(x = time, y = libusage,
color = lib)) +
        geom_point() +
        geom_line() +
        xlab("Every 1000 R scripts from 2008 to 2017") +
        ylab("library frequency counts") +
        ggtitle(paste0("B")) +
        theme(axis.text.x = element_text(angle = 0, hjust = 1),plot.title = e
lement_text(hjust = 0))
```

Code Section 8

```
# check the function usage for each R scripts

load("rfile_list.rda")
pure_text <- lapply(rfile_list, function(x) {x[!grepl("^#", x) & !grepl("\\ ^
#", x) & x != "" & x != " "]})
rfile_text <- unlist(pure_text[!is.na(pure_text)])
# use sys.setlocale to avoid special character usages
Sys.setlocale("LC_ALL", "C")

## [1] "C/C/C/C/C/en_US.UTF-8"

# grep functions out using regular expression
pattern <- gregexpr("[[:alnum:]._ ]+\\(",rfile_text)
functions <- unlist(regmatches(rfile_text,pattern))
functions <- gsub(" *|\\(","",functions)
# count the unique function usage and the frequency of function usage for eac
h unique function
unique_functions <- unique(functions[functions != "library" & functions != "f
unction" & functions != "." & functions != "_" & functions != "" & functions
!= "std"])
function_usage <- sapply(unique_functions, function(x){sum(functions %in% x)}
)
# length of unique function usage
length(unique_functions)

## [1] 10006

# function used 1 or 2 times, which are likely to be self created functions
length(function_usage[function_usage < 3])
```

```
## [1] 7926

# get the top 25 used functions
top_used_functions <- sort(function_usage, decreasing = TRUE)[1:30]
# plot the top 25 used functions for all students
function_dataframe <- data.frame(counts = top_used_functions[1:10],
                                 functions = names(top_used_functions[1:10]))
p4 <- ggplot(function_dataframe, aes(x = reorder(functions,counts), y = count
s)) +
  geom_bar(stat="identity", fill="steelblue") +
  geom_text(aes(y = counts,label = counts), position = position_dodge(width=0
.9),hjust=0,color = "black", size = 2.5) +
  coord_flip() +
  xlab("function") +
  ylab("function frequency counts") +
  ggtitle(paste0("C")) +
  theme(axis.text.x = element_text(angle = 0, hjust = 1),plot.title = element
_text(hjust = 0))
```

Code Section 9

```
# function usage over time

load("rfile_list.rda")
pure_text <- lapply(rfile_list, function(x) {x[!grepl("^#", x) & !grepl("\\ ^
#", x) & x != "" & x != " "]})
pure_text <- pure_text[!is.na(pure_text)]
pure_text_length <- unlist(lapply(pure_text,length))

# if the pure text length is less than 1, we definitely have an empty R scrip
t. We filter those out
pure_text <- pure_text[pure_text_length > 1]
pure_text_length <- unlist(lapply(pure_text,length))
rfile_text <- unlist(pure_text)
unique_function_filter <- top_used_functions



funcusage_matrix <- matrix(data = 0, nrow = length(pure_text),ncol = length(u
nique_function_filter))
for (i in c(1:length(pure_text))){
  row_i <- sapply(names(unique_function_filter), function(x) sum(grepl(paste0
(x,"\\("),pure_text[[i]])))
  funcusage_matrix[i,] <- row_i
}

funcusage_matrix_filter <- funcusage_matrix[,1:10]
funcusage_time <- c()
start <- 1
breaks <- 978
for (i in c(1:20)){
```

```
  end <- start + breaks -1
  if (end > length(pure_text)){
    breaks_sum <- colSums(funcusage_matrix_filter[start:length(pure_text),])
    funcusage_time <- rbind(funcusage_time, breaks_sum)
    break}
  breaks_sum <- colSums(funcusage_matrix_filter[start:end,])
  funcusage_time <- rbind(funcusage_time, breaks_sum)
  start <- end+1
}
colnames(funcusage_time) <- names(unique_function_filter)[1:10]
funcusage_time_vector <- as.vector(funcusage_time)
funcusage_time_df <- data.frame(funcusage = funcusage_time_vector, func = rep
(names(unique_function_filter)[1:10], each = 20), time = rep(c(1:20),10))

function_usage_overtime <- ggplot(funcusage_time_df,aes(x = time, y = funcusa
ge, color = func)) +
        geom_point() +
        geom_line() +
        xlab("Every 1000 R scripts from 2008 to 2017") +
        ylab("function frequency counts") +
        ggtitle(paste0("D")) +
        theme(axis.text.x = element_text(angle = 0, hjust = 1),plot.title = e
lement_text(hjust = 0))

png("library_function.png",width = 1080, height = 700, units = "px")
grid.arrange(p2, library_usage_overtime, p4, function_usage_overtime, ncol =
2, nrow = 2)
dev.off()

## quartz_off_screen
##                 2
```

Code Section 10

```
# build the model and look at the coefficients
model <- lm(log(pure_text_length) ~ ., data = libusage_df)
library_lm_table <- data.frame(coefficients = format(round(model$coefficients
[-1],2), nsmall = 2),
                               p_value = anova(model)$'Pr(>F)'[-31])

# plot the coefficients
coef_dataframe = data.frame(library = names(model$coefficients[!is.na(model$c
oefficients)]), coeff = model$coefficients[!is.na(model$coefficients)])
p3 <- ggplot(coef_dataframe[-1,], aes(x = reorder(library,coeff), y = coeff))
+
  geom_bar(stat="identity", fill="steelblue") +
  coord_flip() +
  xlab("library") +
  ylab("library coefficients") +
  ggtitle("A") +
```

```
  theme(axis.text.x = element_text(angle = 0, hjust = 0, vjust = 0.5, size=16
),plot.title = element_text(hjust = 0))
```

Code Section 11

```
# regression model for function usage #############

unique_function_filter <- names(top_used_functions)
# set the log length of pure text for each student as response, and usage of
each top 25 used function (0 or 1) as predictors
# we build a linear model (lm) using the above response and predictors
funcusage_matrix <- matrix(data = 1, nrow = length(pure_text),ncol = length(u
nique_function_filter))
for (i in c(1:length(pure_text))){
  row_i <- sapply(unique_function_filter, function(x) sum(grepl(paste0(x,"\\(
"),pure_text[[i]])))
  idx <- which(row_i == 0)
  funcusage_matrix[i,idx] <- 0
}

funcusage_df <- as.data.frame(funcusage_matrix)
colnames(funcusage_df) <- c(unique_function_filter)
funcusage_df <- cbind(pure_text_length,funcusage_df)
# build the model and look at the coefficients
model <- lm(log(pure_text_length) ~ ., data = funcusage_df)
function_lm_table <- data.frame(coefficients = format(round(model$coefficient
s[-1],2), nsmall = 2),
                                p_value = anova(model)$'Pr(>F)'[-31])
# we plot the coefficients of the model
coef_dataframe = data.frame(functions = names(model$coefficients[!is.na(model
$coefficients)]), coeff = model$coefficients[!is.na(model$coefficients)])
p5 <- ggplot(coef_dataframe[-1,], aes(x = reorder(functions,coeff), y = coeff
)) +
  geom_bar(stat="identity", fill="steelblue") +
  coord_flip() +
  xlab("function") +
  ylab("function coefficients") +
  ggtitle("B") +
  theme(axis.text.x = element_text(angle = 0, hjust = 0, vjust = 0.5, size=16
),plot.title = element_text(hjust = 0))


png("coefficients.png",width = 1080, height = 700, units = "px")
grid.arrange(p3, p5, ncol = 2)
dev.off()

## quartz_off_screen
##                 2
```

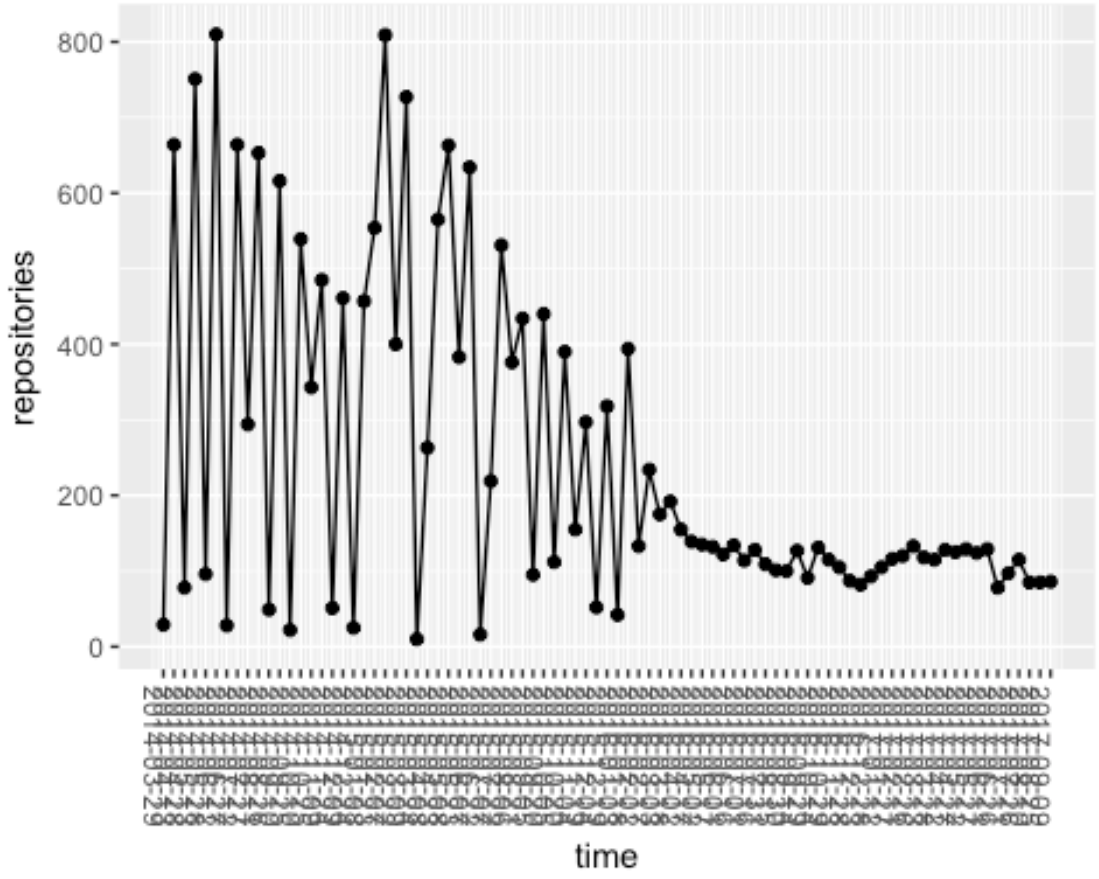
Supplemental Figure

```
print("Supplemental Figure 1")
```

```
## [1] "Supplemental Figure 1"
```

```
p1_repositories
```

Supplemental Figure 1

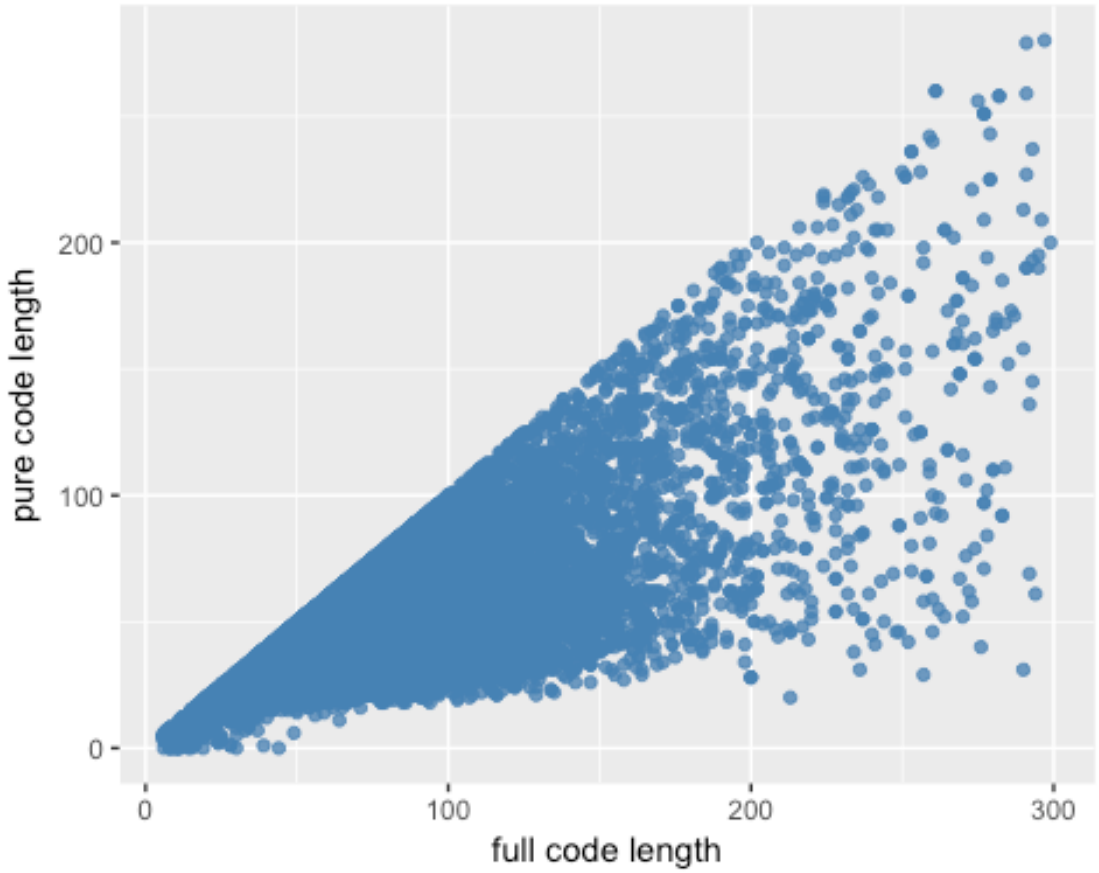



```
print("Supplemental Figure 2")
```

```
## [1] "Supplemental Figure 2"
```

```
p1
```

Supplemental Figure 2




```
print("Supplemental Figure 3")
```

```
## [1] "Supplemental Figure 3"
```

Supplemental Figure 3

library_lm_table

```
##                         coefficients      p_value
## `library(dplyr)`                0.07 3.963739e-37
## `library(plyr)`                 0.02 5.719310e-03
## `library(reshape2)`             0.02 1.433670e-02
## `library(data.table)`           0.08 1.266868e-18
## `library(tidyr)`                0.11 1.488693e-11
## `library(stringr)`              0.21 2.804250e-17
## `library(reshape)`              0.02 5.440353e-01
## `library(sqldf)`                0.26 5.538828e-11
## `library(readr)`                0.20 1.925523e-04
## `library(downloader)`           0.14 3.195074e-03
## `library(knitr)`                0.10 3.743138e-02
## `library(utils)`                0.24 4.471825e-07
## `library(RCurl)`                0.11 7.360942e-02
## `library(httr)`                 0.03 2.348636e-01
```

```
## `library(Hmisc)`               0.26 6.313104e-05
## `library(XML)`                 0.04 8.967550e-01
## `library(gdata)`               0.20 6.005029e-03
## `library(xlsx)`               -0.05 7.330255e-01
## `library(LaF)`                 0.41 2.378919e-07
## `library(lubridate)`           0.19 3.036215e-02
## `library(stats)`               0.16 8.455752e-02
## `library(car)`                -0.02 4.919203e-01
## `library(magrittr)`            0.24 1.886618e-02
## `library(tools)`               0.80 5.643353e-14
## `library(memisc)`              0.19 8.589096e-02
## `library(qdap)`                0.08 5.165150e-01
## `library(stringi)`             0.12 3.056594e-01
## `library(base)`                0.16 2.230654e-01
## `library(tidyverse)`          -0.03 7.860702e-01
## `library(doBy)`                0.04 7.570424e-01
```

```
print("Supplemental Figure 4")
```

```
## [1] "Supplemental Figure 4"
```

Supplemental Figure 3

function_lm_table

```
##                coefficients       p_value
## read.table             0.17 3.857176e-118
## names                  0.35 1.440293e-302
## c                      0.18 9.307596e-164
## gsub                  -0.04 3.962386e-113
## colnames               0.02  1.191499e-01
## rbind                  0.08  1.986645e-01
## cbind                  0.01  5.894743e-01
## mean                   0.23 2.161656e-314
## write.table            0.02  7.419460e-03
## paste                  0.24  0.000000e+00
## grep                  -0.05  2.028060e-21
## `if`                   0.10 1.592833e-152
## grepl                  0.01  1.072039e-01
## file.path              0.16  6.314697e-35
## rm                     0.10  2.324845e-50
## as.character           0.00  5.088769e-02
## `for`                  0.18  3.387060e-92
## file.exists            0.13  4.551401e-51
## print                  0.28 3.429822e-147
## merge                  0.06  1.374291e-15
## sub                    0.18  2.866525e-31
## setwd                  0.04  3.951748e-10
## dim                    0.18  4.030393e-83
```

```
## length                0.19  9.463314e-91
## read.csv              0.16  2.075585e-25
## download.file         0.00  7.836587e-01
## unzip                -0.01  8.572120e-01
## aggregate            -0.03  1.324749e-09
## factor                0.01  1.660302e-01
## select                0.13  4.193893e-39
```

Reference:

[1] Hadley Wickham and Romain Francois (2016). dplyr: A Grammar of Data Manipulation. R package version 0.5.0. https://CRAN.R-project.org/package=dplyr

[2] Jennifer Bryan and Hadley Wickham (NA). gh: 'GitHub' 'API'. R package version 1.0.1.https://github.com/r-lib/gh#readme

[3] Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL http://www.jstatsoft.org/v40/i03

[4] H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2009

[5] Baptiste Auguie (2016). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.2.1. https://CRAN.R-project.org/package=gridExtra

[6] Sarkar, Deepayan (2008) Lattice: Multivariate Data Visualization with R. Springer, New York. ISBN 978-0-387-75968-5