Today, we’ll show how to work with this database in R and the {tidyverse} to paint our own pictures of application usage.

There are quite a number of tables in the knowledgeC.db SQLite 3 database:



That visual schema was created in OmniGraffle via a below script that uses the OmniGraffle automation framework. The OmniGraffle source files are also available upon request.

library(tidyverse)

omnijs <- function(cmd, wait = 0.10) {

browseURL(

sprintf("omnigraffle:///omnijs-run?script=%s", URLencode(cmd))

)

}

omni\_schema\_src <- function(src) {

db\_tbls <- src\_tbls(src)

start <- 0

max\_len <- 0

sz <- 9

space\_over <- 10 \* sz

map\_chr(db\_tbls, function(tbl\_name) {

suppressWarnings(

tbl(src, tbl\_name) %>%

head(1) %>%

collect() %>%

as.data.frame() %>%

summarise\_all(pillar::new\_pillar\_type) %>%

gather(variable, class) -> schema

)

rows <- as.character(glue::glue('{schema$variable} <{schema$class}>'))

map\_chr(1:nrow(schema), ~{

as.character(glue::glue("\ntable.graphicAt({.x}, 0).text = '{rows[.x]}'\n"))

}) %>%

paste0(collapse = "\n") -> sch

if (start != 0) start <<- start + space\_over

max\_len <<- max(nchar(rows))

glue::glue("

cnvs = document.windows[0].selection.canvas

rect = new Rect({start}, 0, {max\_len} \* {sz}, 26)

graphic = cnvs.addShape('Rectangle', rect)

// graphic properties

graphic.strokeThickness = 0.25

graphic.strokeColor = Color.black

// solid properties

graphic.fillColor = Color.white

graphic.fillType = FillType.Solid

// solid text properties

graphic.text = ''

graphic.textSize = 12

graphic.textColor = Color.black

graphic.textHorizontalAlignment = HorizontalTextAlignment.Left

graphic.textVerticalPlacement = VerticalTextPlacement.Middle

graphic.fontName = 'Monaco'

graphic.textHorizontalPadding = 10

graphic.textVerticalPadding = 0

// create the base table

table = new Table(graphic)

table.columns = 1

table.rows = {nrow(schema) + 1}

// adjust header row and column size

table.setRowHeight(36, 0)

table.graphicAt(0, 0).text = '{tbl\_name}'

table.graphicAt(0, 0).textHorizontalAlignment = HorizontalTextAlignment.Center

table.graphicAt(0, 0).fillColor = Color.black

table.graphicAt(0, 0).textColor = Color.white

table.graphicAt(0, 0).textSize = 14

{sch}

") %>% as.character() -> out

start <<- start + (max\_len \* sz)

out

}) -> schema\_out

omnijs(paste0(schema\_out, collapse = "\n\n"))

}

src\_sqlite("~/Library/Application Support/Knowledge/knowledgeC.db") %>%

omni\_schema\_src()



Most of the interesting bits (for any tracking-related spelunking) are in the ZOBJECT table and to get a full picture of usage we’ll need to join it with some other tables that are connected via a few foreign keys:

There are a few ways to do this in {tidyverse} R. The first is an extended straight SQL riff off of one of Sarah’s original queries:

library(hrbrthemes) # for ggplot2 machinations

library(tidyverse)

# source the knowledge db

kdb <- src\_sqlite("~/Library/Application Support/Knowledge/knowledgeC.db")

tbl(

kdb,

sql('

SELECT

ZOBJECT.ZVALUESTRING AS "app",

(ZOBJECT.ZENDDATE - ZOBJECT.ZSTARTDATE) AS "usage",

CASE ZOBJECT.ZSTARTDAYOFWEEK

WHEN "1" THEN "Sunday"

WHEN "2" THEN "Monday"

WHEN "3" THEN "Tuesday"

WHEN "4" THEN "Wednesday"

WHEN "5" THEN "Thursday"

WHEN "6" THEN "Friday"

WHEN "7" THEN "Saturday"

END "dow",

ZOBJECT.ZSECONDSFROMGMT/3600 AS "tz",

DATETIME(ZOBJECT.ZSTARTDATE + 978307200, \'UNIXEPOCH\') as "start\_time",

DATETIME(ZOBJECT.ZENDDATE + 978307200, \'UNIXEPOCH\') as "end\_time",

DATETIME(ZOBJECT.ZCREATIONDATE + 978307200, \'UNIXEPOCH\') as "created\_at",

CASE ZMODEL

WHEN ZMODEL THEN ZMODEL

ELSE "Other"

END "source"

FROM

ZOBJECT

LEFT JOIN

ZSTRUCTUREDMETADATA

ON ZOBJECT.ZSTRUCTUREDMETADATA = ZSTRUCTUREDMETADATA.Z\_PK

LEFT JOIN

ZSOURCE

ON ZOBJECT.ZSOURCE = ZSOURCE.Z\_PK

LEFT JOIN

ZSYNCPEER

ON ZSOURCE.ZDEVICEID = ZSYNCPEER.ZDEVICEID

WHERE

ZSTREAMNAME = "/app/usage"'

)) -> usage

usage

## # Source: SQL [?? x 8]

## # Database: sqlite 3.29.0 [/Users/johndoe/Library/Application Support/Knowledge/knowledgeC.db]

## app usage dow tz start\_time end\_time created\_at source

##

## 1 com.bitrock.appinstaller 15 Friday -4 2019-10-05 01:11:27 2019-10-05 01:11:42 2019-10-05 01:11:… MacBookPro13…

## 2 com.tinyspeck.slackmacg… 4379 Tuesday -4 2019-10-01 13:19:24 2019-10-01 14:32:23 2019-10-01 14:32:… Other

## 3 com.tinyspeck.slackmacg… 1167 Tuesday -4 2019-10-01 18:19:24 2019-10-01 18:38:51 2019-10-01 18:38:… Other

## 4 com.tinyspeck.slackmacg… 1316 Tuesday -4 2019-10-01 19:13:49 2019-10-01 19:35:45 2019-10-01 19:35:… Other

## 5 com.tinyspeck.slackmacg… 12053 Thursday -4 2019-10-03 12:25:18 2019-10-03 15:46:11 2019-10-03 15:46:… Other

## 6 com.tinyspeck.slackmacg… 1258 Thursday -4 2019-10-03 15:50:16 2019-10-03 16:11:14 2019-10-03 16:11:… Other

## 7 com.tinyspeck.slackmacg… 2545 Thursday -4 2019-10-03 16:24:30 2019-10-03 17:06:55 2019-10-03 17:06:… Other

## 8 com.tinyspeck.slackmacg… 303 Thursday -4 2019-10-03 17:17:10 2019-10-03 17:22:13 2019-10-03 17:22:… Other

## 9 com.tinyspeck.slackmacg… 9969 Thursday -4 2019-10-03 17:33:38 2019-10-03 20:19:47 2019-10-03 20:19:… Other

## 10 com.tinyspeck.slackmacg… 2813 Thursday -4 2019-10-03 20:19:52 2019-10-03 21:06:45 2019-10-03 21:06:… Other

## # … with more rows

Before explaining what that query does, let’s rewrite it {dbplyr}-style:

tbl(kdb, "ZOBJECT") %>%

mutate(

created\_at = datetime(ZCREATIONDATE + 978307200, "UNIXEPOCH", "LOCALTIME"),

start\_dow = case\_when(

ZSTARTDAYOFWEEK == 1 ~ "Sunday",

ZSTARTDAYOFWEEK == 2 ~ "Monday",

ZSTARTDAYOFWEEK == 3 ~ "Tuesday",

ZSTARTDAYOFWEEK == 4 ~ "Wednesday",

ZSTARTDAYOFWEEK == 5 ~ "Thursday",

ZSTARTDAYOFWEEK == 6 ~ "Friday",

ZSTARTDAYOFWEEK == 7 ~ "Saturday"

),

start\_time = datetime(ZSTARTDATE + 978307200, "UNIXEPOCH", "LOCALTIME"),

end\_time = datetime(ZENDDATE + 978307200, "UNIXEPOCH", "LOCALTIME"),

usage = (ZENDDATE - ZSTARTDATE),

tz = ZSECONDSFROMGMT/3600

) %>%

left\_join(tbl(kdb, "ZSTRUCTUREDMETADATA"), c("ZSTRUCTUREDMETADATA" = "Z\_PK")) %>%

left\_join(tbl(kdb, "ZSOURCE"), c("ZSOURCE" = "Z\_PK")) %>%

left\_join(tbl(kdb, "ZSYNCPEER"), "ZDEVICEID") %>%

filter(ZSTREAMNAME == "/app/usage") %>%

select(

app = ZVALUESTRING, created\_at, start\_dow, start\_time, end\_time, usage, tz, source = ZMODEL

) %>%

mutate(source = ifelse([is.na](http://is.na)(source), "Other", source)) %>%

collect() %>%

mutate\_at(vars(created\_at, start\_time, end\_time), as.POSIXct) -> usage

What we’re doing is pulling out the day of week, start/end usage times & timezone info, app bundle id, source of the app interactions and the total usage time for each entry along with when that entry was created. We need to do some maths since Apple stores time-y whime-y info in its own custom format, plus we need to convert numeric DOW to labeled DOW.

The bundle ids are pretty readable, but they’re not really intended for human consumption, so we’ll make a translation table for the bundle id to app name by using the [mdls](https://ss64.com/osx/mdls.html) command.

list.files(

c("/Applications", "/System/Library/CoreServices", "/Applications/Utilities", "/System/Applications"), # main places apps are stored (there are potentially more but this is sufficient for our needs)

pattern = "\\.app$",

full.names = TRUE

) -> apps

x <- sys::exec\_internal("mdls", c("-name", "kMDItemCFBundleIdentifier", "-r", apps))

# mdls null (\0) terminates each entry so we have to do some raw surgery to get it into a format we can use

x$stdout[x$stdout == as.raw(0)] <- as.raw(0x0a)

tibble(

name = gsub("\\.app$", "", basename(apps)),

app = read\_lines(x$stdout)

) -> app\_trans

app\_trans

## # A tibble: 270 x 2

## name app

##

## 1 1Password 7 com.agilebits.onepassword7

## 2 Adium com.adiumX.adiumX

## 3 Agenda com.momenta.agenda.macos

## 4 Alfred 4 com.runningwithcrayons.Alfred

## 5 Amazon Music com.amazon.music

## 6 Android File Transfer com.google.android.mtpviewer

## 7 Awsaml com.rapid7.awsaml

## 8 Bartender 2 com.surteesstudios.Bartender

## 9 BBEdit com.barebones.bbedit

## 10 BitdefenderVirusScanner com.bitdefender.BitdefenderVirusScanner

## # … with 260 more rows

The usage info goes back ~30 days, so let’s do a quick summary of the top 10 apps and their total usage (in hours):

usage %>%

group\_by(app) %>%

summarise(first = min(start\_time), last = max(end\_time), total = sum(usage, na.rm=TRUE)) %>%

ungroup() %>%

mutate(total = total / 60 / 60) %>% # hours

arrange(desc(total)) %>%

left\_join(app\_trans) -> overall\_usage

overall\_usage %>%

slice(1:10) %>%

left\_join(app\_trans) %>%

mutate(name = fct\_inorder(name) %>% fct\_rev()) %>%

ggplot(aes(x=total, y=name)) +

geom\_segment(aes(xend=0, yend=name), size=5, color = ft\_cols$slate) +

scale\_x\_comma(position = "top") +

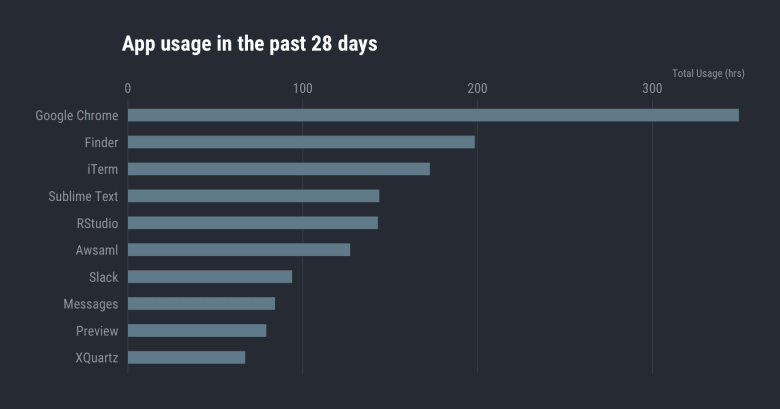
labs(

x = "Total Usage (hrs)", y = NULL,

title = glue::glue('App usage in the past {round(as.numeric(max(usage$end\_time) - min(usage$start\_time), "days"))} days')

) +

theme\_ft\_rc(grid="X")

There’s a YUGE flaw in the current way macOS tracks application usage. Unlike iOS where apps really don’t run simultaneously (with iPadOS they *kinda* can/do, now), macOS apps are usually started and left open along with other apps. Apple doesn’t do a great job identifying only active app usage activity so many of these usage numbers are heavily inflated. Hopefully that will be fixed by macOS 10.15.

We have more data at our disposal, so let’s see when these apps get used. To do that, we’ll use segments to plot individual usage tracks and color them by weekday/weekend usage (still limiting to top 10 for blog brevity):

usage %>%

filter(app %in% overall\_usage$app[1:10]) %>%

left\_join(app\_trans) %>%

mutate(name = factor(name, levels = rev(overall\_usage$name[1:10]))) %>%

ggplot() +

geom\_segment(

aes(

x = start\_time, xend = end\_time, y = name, yend = name,

color = ifelse(start\_dow %in% c("Saturday", "Sunday"), "Weekend", "Weekday")

),

size = 10,

) +

scale\_x\_datetime(position = "top") +

scale\_colour\_manual(

name = NULL,

values = c(

"Weekend" = ft\_cols$light\_blue,

"Weekday" = ft\_cols$green

)

) +

guides(

colour = guide\_legend(override.aes = list(size = 1))

) +

labs(

x = NULL, y = NULL,

title = glue::glue('Top 10 App usage on this Mac in the past {round(as.numeric(max(usage$end\_time) - min(usage$start\_time), "days"))} days'),

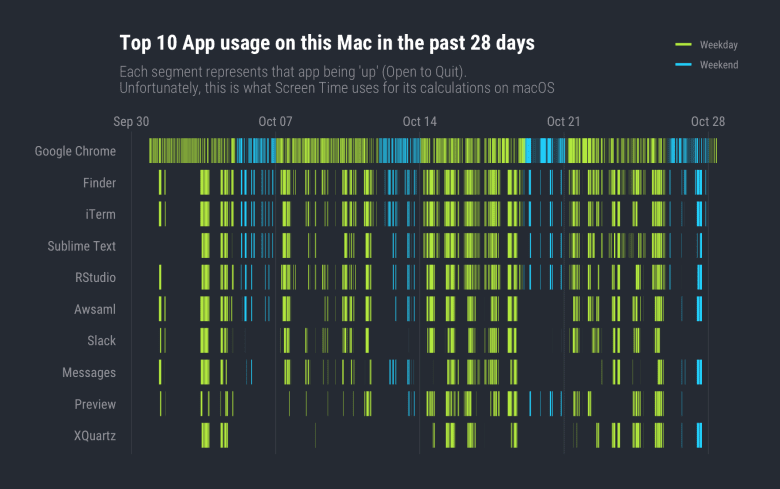
subtitle = "Each segment represents that app being 'up' (Open to Quit).\nUnfortunately, this is what Screen Time uses for its calculations on macOS"

) +

theme\_ft\_rc(grid="X") +

theme(legend.position = c(1, 1.25)) +

theme(legend.justification = "right")



I’m not entirely sure “on this Mac” is completely accurate since I think this syncs across all active Screen Time devices due to this (*n* is in seconds):

count(usage, source, wt=usage, sort=TRUE)

## # A tibble: 2 x 2

## source n

##

## 1 Other 4851610

## 2 MacBookPro13,2 1634137

The “Other” appears to be the work-dev Mac but it doesn’t have the identifier mapped so I think that means it’s the local one and that the above chart is looking at Screen Time across all devices. I literally (right before this sentence) enabled Screen Time on my iPhone so we’ll see if that ends up in the database and I’ll post a quick update if it does.

We’ll take one last look by day of week and use a heatmap to see the results:

count(usage, start\_dow, app, wt=usage/60/60) %>%

left\_join(app\_trans) %>%

filter(app %in% overall\_usage$app[1:10]) %>%

mutate(name = factor(name, levels = rev(overall\_usage$name[1:10]))) %>%

mutate(start\_dow = factor(start\_dow, c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))) %>%

ggplot() +

geom\_tile(aes(start\_dow, name, fill = n), color = "#252a32", size = 0.75) +

scale\_x\_discrete(expand = c(0, 0.5), position = "top") +

scale\_y\_discrete(expand = c(0, 0.5)) +

scale\_fill\_viridis\_c(direction = -1, option = "magma", name = "Usage (hrs)") +

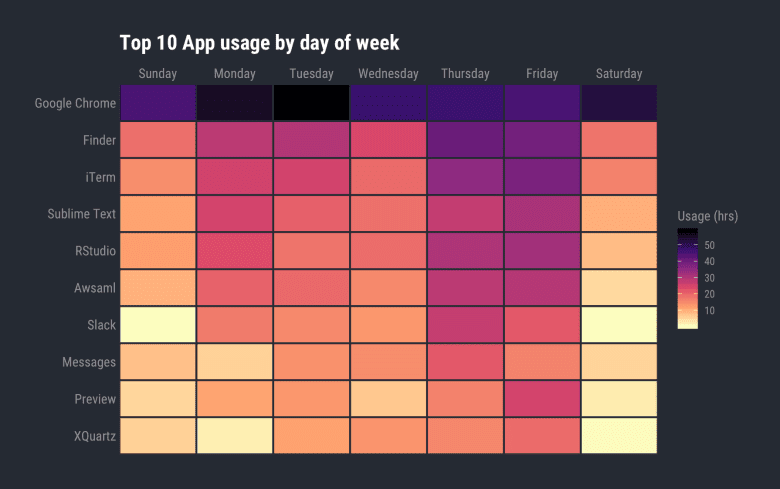
labs(

x = NULL, y = NULL,

title = "Top 10 App usage by day of week"

) +

theme\_ft\_rc(grid="")



**FIN**

Apple’s Screen Time also tracks “category”, which is something we can pick up from each application’s embedded metadata. We’ll do that in a follow-up post along with seeing whether we can capture iOS usage now that I’ve enabled Screen Time on those devices as well.

Keep spelunking the knowledgeC.db table(s) and blog about or reply in the comments with any interesting nuggets you find.