Welcome to Our House on Maple Avenue:

an overview of Toronto's municipally-owned trees and the patterns within them

due February 6, 2022*

Abstract

This paper examines a dataset from the Open Data Toronto portal, cataloguing the more than 660,000 trees owned by the City across its 25 electoral wards. Through careful massaging and interpretation of this data, we discover correlative trends between species of trees, their assorted sizes, and even the names of streets on which they were planted. We also examine whether any biases or patterns emerge in regards to a tree's location in one of Toronto's four Community Council Areas: Toronto East York, North York, Etobicoke York, and Scarborough. As one of Upper Canada's oldest cities, it may be unsurprising that the plurality of Toronto's municipal trees are maples, planted on Avenues; what else can we discover from these data?

Contents

Introduction

[TODO] Lorem markdownum nocuisse attonitos spectarat, et caede aut *limine iussit* intrare carinae tumulandus bracchia cavo cum olivae sede. **Voces Lydas communis** ignara subvolat quoque: annis Antiphatae tenet!

1. Recepi eat usque linguisque qua nescia leges
2. Mihi arces circumtulit pocula cedit et ter
3. Sono Iunonem Autonoe

Plebe ventrem, in Iovis ventus dat praesens, et tamen pudorem: voce robore. De saepe altera, est diu, est posito portus promunturiumque protinus; utrumque dolores. Harundine iamque, odio, est, loco, teque, aether opus cycnorum, sua bellum, sum. Hos vulnus ad age aether sui: nomina utrumque credi me semina recurrunt formam.

Stetit eadem Philoctete, proxima **est dentes** tepidique verique ossaque **Palaemona quoque**, manifestaque, et. Inmanem insula Ilion: inque dote sidera inspirantque patruo maiestatemque

*Extended

magni vices: erat. Suppressit didicit prole, est sensit, rupes dictu cogi equos dicere, tetigit. Fuisse deploratosque sponte et caelo quamvis.

Aqua fugiant redditur armenta atque parentem prece vix tua ausi! Quo euntis opes nymphas simulat suis. Vacuos quod quod locis Libycas mihi, e protexi lumina sacrilegi densi Ilioneus animalia; *sua posset difficile*. Timide unus Phoebi suscitat.

- Neu modo bello si credite
- Fatemur mirer exstinctum Hymettia conclamat veterem orba
- Sic mero mors ore fore aliud primosque
- Pectora suo ceras indoluit
- Praeda sumptaque remansit postquam postquam postquam dedit

Telique dolore; vidit undas; rudis barbarus vestras murice **vestras**. **Virorum** evocat.

Lapsu praeceps rapta puerilibus surgit, tellus illa aderam vixisse non. Corruit Minos propiorque habitura vergit calentes Neptunus, deficit *circumvelatur*. Leve ab utraque nec petis et vocibus sagittas veteris in dicit si? Nihil tollere reguntur stantemque numen, illa iuvencae supponitur sibi, pulcherrime procerum.

In addition to `knitr` (Xie 2021) for generating this paper, `knitr::kable` and `kableExtra` (Zhu 2021) were indispensable for tabular formatting. And, of course, we are thoroughly indebted to Alexander (2022)'s book and associated course, Telling Stories with Data for instruction on how to put all the parts together into a cogent paper.

Data

All data were analyzed using the R statistical programming language (R Core Team 2021). This research was primarily facilitated by the RStudio IDE (RStudio Team 2022) for easy combination of visual analysis, ad-hoc batch-scripting in a REPL environment, and then the authoring of both data-massaging scripts and this paper via R Markdown. Upon first cloning the GitHub repository[1] and running setup scripts, data are fetched in CSV format via a simple HTTP request to the Open Data Toronto portal. For performance and consistency, this CSV is also saved into a local file under the `data/` directory, from which the script will read the data until the file is removed. A number of additional libraries and packages were included for researcher quality-of-life and performance purposes, all of which fall under the familiar umbrella of `tidyverse` (Wickham et al. 2019). However, two subsidiary packages, namely `dplyr` (Wickham et al. 2021) and `stringr` (Wickham 2019), were also explicitly imported into the global namespace. Visualization and graphing are handled by the excellent `ggplot2` (Wickham 2016).

On the Open Data Portal proper, the Division of Parks, Forestry & Recreation, who publish the dataset, describe its contents as "primarily pertain[ing] to City-owned trees located on road allowances across Toronto… compiled by Urban Forestry[2] staff during their inspections or tree maintenance work" (Parks, Forestry & Recreation 2022). However, the description also outlines some limitations of the dataset: as tree data is only updated alongside the Urban Forestry branch's line of work, some data "may be several years old and… there may be inaccuracies with the data as a result." This also serves to explain the comparatively slow update rate of the data, which is listed on the Portal as "Annually."

[1] Data and code can be found at https://github.com/oliver-daniel/inf_313-paper_1.
[2] A branch of Parks, Forestry & Recreation.

Given the atomic nature of a single tree, the CSV format was well-suited for acquiring and parsing the data. Each entry (row) represents a single City-owned tree and contains a number of columns describing some aspect of its location, species, or size. Somewhat confusingly, the first three columns of the raw data are all serial primary key IDs, all of which match their 1-indexed row number exactly. The provided data specification (Appendix; Figure 1) provides three distinct purposes for each ID, at least one of which is City-centric; however, as there were no anomalous entries, these columns were dropped during data preparation, and simple row number sufficed when a primary key was needed. Ultimately, the four retained columns were as follows:

1. `STREETNAME`: The street along which the tree was planted;
2. `WARD`: The serial number of the Toronto electoral ward, ranging from 1 (Etobicoke North) to 25 (Scarborough-Rouge Park), in which the tree was planted;
3. `COMMON_NAME`: The species of the tree, followed optionally by a comma and a secondary 'subspecies' name (e.g., 'Maple' or 'Maple, Norway');
4. `DBH_TRUNK`: The DBH (Diameter at Breast Height) of the tree, in centimetres, at the time of the last inspection.

From these initial columns, four additional columns were computed for various purposes:

1. `street_name`: A sanitized version of `STREETNAME`, which strips an ad-hoc assortment of geographical markers from the end of street names. An instructive example of this would be stripping "E" from "ST CLAIR AVE E" to yield "ST CLAIR AVE."
2. `street_suffix`: The road affordance suffix: "AVE," "ST," "PK," etc. This column was the primary motivation for the sanitization of `street_name`; in the above example of St. Clair, this value would be "AVE" rather than "E."
3. `district`: The Community Council Area[3] in which `WARD` resides, one of four. "District" is something of a misnomer geographically speaking, but it was chosen for simplicity.
4. `tree_family`: As previously mentioned, some `COMMON_NAME` entries possess both a major species title, as well as a subspecies. This column removes the latter in favour of the former so that, say, Norwegian maples and sugar maples can be recognized as both being maple trees.

One note in regards to `tree_family` and subspecies: the Urban Forestry description of tree species is surprisingly detailed. Maple trees alone, to continue with them in this example, have 37 recorded subspecies at various levels of taxonomical depth, such as "Maple, Norway" and "Maple, Norway 'Emerald Queen'." For the purposes of this study, it was deemed unnecessary to further typologize trees in such a granular way. Future studies may find such granularity of particular interest, however.

Discussion

[3]Based on the new Community Council boundaries from December 2018: https://www.toronto.ca/city-government/data-research-maps/neighbourhoods-communities/community-council-area-profiles/

Appendix

Column	Description
_id	Unique row identifier for Open Data database
OBJECTID	Object identifier
STRUCTID	Unique numeric id for tree asset, for City use
ADDRESS	Street number adjacent to the asset (ID: 5100)
STREETNAME	The name of the street adjacent to the asset (IE: Yonge St)
CROSSSTREET1	Cross street 1 nearest the address (IE: Hillcrest Ave)
CROSSSTREET2	Cross street 1 nearest the address (IE: Empress Ave)
SUFFIX	is the word that follows the name of a street to further describe that street (IE: A)
UNIT_NUMBER	Represent the Unit / Apartment or Suite Number at the address
TREE_POSITION_NUMBER	Coding for multiple trees on a site, the first tree will be in position 10 (going in the same direction that addresses increase on the street).
SITE	Park name (IE: High Park)
WARD	Municipal Ward the address falls in (IE: 25)
COMMON_NAME	Common name for tree species
DBH_TRUNK	Measurement of diameter at breast height (1.3 m)
geometry	Tree location geometry

Figure 1: Column data specification, provided by the Open Data Toronto Portal.

```
## function (..., list = character(), package = NULL, lib.loc = NULL,
##     verbose = getOption("verbose"), envir = .GlobalEnv, overwrite = TRUE)
## {
##     fileExt <- function(x) {
##         db <- grepl("\\.[^.]+\\.(gz|bz2|xz)$", x)
##         ans <- sub(".*\\.", "", x)
##         ans[db] <- sub(".*\\.([^.]+\\.)(gz|bz2|xz)$", "\\1\\2",
##             x[db])
##         ans
##     }
##     my_read_table <- function(...) {
##         lcc <- Sys.getlocale("LC_COLLATE")
##         on.exit(Sys.setlocale("LC_COLLATE", lcc))
##         Sys.setlocale("LC_COLLATE", "C")
##         read.table(...)
```

```
##     }
##     names <- c(as.character(substitute(list(...))[-1L]), list)
##     if (!is.null(package)) {
##         if (!is.character(package))
##             stop("'package' must be a character string or NULL")
##         if (FALSE) {
##             if (any(package %in% "base"))
##                 warning("datasets have been moved from package 'base' to package 'datasets'")
##             if (any(package %in% "stats"))
##                 warning("datasets have been moved from package 'stats' to package 'datasets'")
##             package[package %in% c("base", "stats")] <- "datasets"
##         }
##     }
##     paths <- find.package(package, lib.loc, verbose = verbose)
##     if (is.null(lib.loc))
##         paths <- c(path.package(package, TRUE), if (!length(package)) getwd(),
##             paths)
##     paths <- unique(normalizePath(paths[file.exists(paths)]))
##     paths <- paths[dir.exists(file.path(paths, "data"))]
##     dataExts <- tools:::.make_file_exts("data")
##     if (length(names) == 0L) {
##         db <- matrix(character(), nrow = 0L, ncol = 4L)
##         for (path in paths) {
##             entries <- NULL
##             packageName <- if (file_test("-f", file.path(path,
##                 "DESCRIPTION")))
##                 basename(path)
##             else "."
##             if (file_test("-f", INDEX <- file.path(path, "Meta",
##                 "data.rds"))) {
##                 entries <- readRDS(INDEX)
##             }
##             else {
##                 dataDir <- file.path(path, "data")
##                 entries <- tools::list_files_with_type(dataDir,
##                   "data")
##                 if (length(entries)) {
##                   entries <- unique(tools::file_path_sans_ext(basename(entries)))
##                   entries <- cbind(entries, "")
##                 }
##             }
##             if (NROW(entries)) {
##                 if (is.matrix(entries) && ncol(entries) == 2L)
##                   db <- rbind(db, cbind(packageName, dirname(path),
##                     entries))
##                 else warning(gettextf("data index for package %s is invalid and will be ignore
##                   sQuote(packageName)), domain = NA, call. = FALSE)
##             }
```

```
##         }
##         colnames(db) <- c("Package", "LibPath", "Item", "Title")
##         footer <- if (missing(package))
##             paste0("Use ", sQuote(paste("data(package =", ".packages(all.available = TRUE))")),
##                 "\n", "to list the data sets in all *available* packages.")
##         else NULL
##         y <- list(title = "Data sets", header = NULL, results = db,
##             footer = footer)
##         class(y) <- "packageIQR"
##         return(y)
##     }
##     paths <- file.path(paths, "data")
##     for (name in names) {
##         found <- FALSE
##         for (p in paths) {
##             tmp_env <- if (overwrite)
##                 envir
##             else new.env()
##             if (file_test("-f", file.path(p, "Rdata.rds"))) {
##                 rds <- readRDS(file.path(p, "Rdata.rds"))
##                 if (name %in% names(rds)) {
##                   found <- TRUE
##                   if (verbose)
##                     message(sprintf("name=%s:\t found in Rdata.rds",
##                       name), domain = NA)
##                   thispkg <- sub(".*/([^/]*)/data$", "\\1", p)
##                   thispkg <- sub("_.*$", "", thispkg)
##                   thispkg <- paste0("package:", thispkg)
##                   objs <- rds[[name]]
##                   lazyLoad(file.path(p, "Rdata"), envir = tmp_env,
##                     filter = function(x) x %in% objs)
##                   break
##                 }
##                 else if (verbose)
##                   message(sprintf("name=%s:\t NOT found in names() of Rdata.rds, i.e.,\n\t%s\n
##                     name, paste(names(rds), collapse = ",")),
##                     domain = NA)
##             }
##             if (file_test("-f", file.path(p, "Rdata.zip"))) {
##                 warning("zipped data found for package ", sQuote(basename(dirname(p))),
##                   ".\nThat is defunct, so please re-install the package.",
##                   domain = NA)
##                 if (file_test("-f", fp <- file.path(p, "filelist")))
##                   files <- file.path(p, scan(fp, what = "", quiet = TRUE))
##                 else {
##                   warning(gettextf("file 'filelist' is missing for directory %s",
##                     sQuote(p)), domain = NA)
##                   next
```

```
##                 }
##             }
##             else {
##                 files <- list.files(p, full.names = TRUE)
##             }
##             files <- files[grep(name, files, fixed = TRUE)]
##             if (length(files) > 1L) {
##                 o <- match(fileExt(files), dataExts, nomatch = 100L)
##                 paths0 <- dirname(files)
##                 paths0 <- factor(paths0, levels = unique(paths0))
##                 files <- files[order(paths0, o)]
##             }
##             if (length(files)) {
##                 for (file in files) {
##                   if (verbose)
##                     message("name=", name, ":\t file= ...", .Platform$file.sep,
##                       basename(file), "::\t", appendLF = FALSE,
##                       domain = NA)
##                   ext <- fileExt(file)
##                   if (basename(file) != paste0(name, ".", ext))
##                     found <- FALSE
##                   else {
##                     found <- TRUE
##                     zfile <- file
##                     zipname <- file.path(dirname(file), "Rdata.zip")
##                     if (file.exists(zipname)) {
##                       Rdatadir <- tempfile("Rdata")
##                       dir.create(Rdatadir, showWarnings = FALSE)
##                       topic <- basename(file)
##                       rc <- .External(C_unzip, zipname, topic,
##                         Rdatadir, FALSE, TRUE, FALSE, FALSE)
##                       if (rc == 0L)
##                         zfile <- file.path(Rdatadir, topic)
##                     }
##                     if (zfile != file)
##                       on.exit(unlink(zfile))
##                     switch(ext, R = , r = {
##                       library("utils")
##                       sys.source(zfile, chdir = TRUE, envir = tmp_env)
##                     }, RData = , rdata = , rda = load(zfile,
##                       envir = tmp_env), TXT = , txt = , tab = ,
##                       tab.gz = , tab.bz2 = , tab.xz = , txt.gz = ,
##                       txt.bz2 = , txt.xz = assign(name, my_read_table(zfile,
##                         header = TRUE, as.is = FALSE), envir = tmp_env),
##                       CSV = , csv = , csv.gz = , csv.bz2 = ,
##                       csv.xz = assign(name, my_read_table(zfile,
##                         header = TRUE, sep = ";", as.is = FALSE),
##                         envir = tmp_env), found <- FALSE)
```

```
##                   }
##                   if (found)
##                     break
##                 }
##                 if (verbose)
##                   message(if (!found)
##                     "*NOT* ", "found", domain = NA)
##             }
##             if (found)
##                 break
##         }
##         if (!found) {
##             warning(gettextf("data set %s not found", sQuote(name)),
##                 domain = NA)
##         }
##         else if (!overwrite) {
##             for (o in ls(envir = tmp_env, all.names = TRUE)) {
##                 if (exists(o, envir = envir, inherits = FALSE))
##                   warning(gettextf("an object named %s already exists and will not be overwrit
##                     sQuote(o)))
##                 else assign(o, get(o, envir = tmp_env, inherits = FALSE),
##                   envir = envir)
##             }
##             rm(tmp_env)
##         }
##     }
##     invisible(names)
## }
## <bytecode: 0x55907e02eb68>
## <environment: namespace:utils>
```

References

Alexander, Rohan. 2022. *Telling Stories with Data*. https://www.tellingstorieswithdata.com.

Parks, Forestry & Recreation. 2022. *Street Tree Data*. Open Data Toronto. https://open.toronto.ca/dataset/street-tree-data.

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

RStudio Team. 2022. *RStudio: Integrated Development Environment for r*. Boston, MA: RStudio, PBC. http://www.rstudio.com/.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2019. *Stringr: Simple, Consistent Wrappers for Common String Operations*. https://CRAN.R-project.org/package=stringr.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2021. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Xie, Yihui. 2021. *Knitr: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.org/knitr/.

Zhu, Hao. 2021. *kableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.