

# Epidémiologie mathématique à l'heure du “Open Data”

Julien Arino

Department of Mathematics  
Data Science Nexus  
Visual and Automatic Disease Analytics training program  
Centre for Disease Modelling (West)

University of Manitoba

[Julien.Arino@umanitoba.ca](mailto:Julien.Arino@umanitoba.ca)

26 novembre 2019

## Remarques sur ce document

Tous les liens devraient être clickables

Le code sera fourni (sur demande)

Le code est en R .. j'aurais pu utiliser Python mais je hais ce language :)

Une partie de ces transparents est dynamique: ils sont produits avec Rmarkdown en utilisant des données extraites du web et à jour au jour de compilation de ce document (2022-05-01)

## Pour générer ces transparents

Il vous faudra les programmes (gratuits) suivants

- R (une version récente  $\geq 3.5$ ) (lien)
- RStudio (lien)
- Une distribution de L<sup>A</sup>T<sub>E</sub>X(MiK<sup>T</sup>eX sous Windows, TeX Live sous Linux & Mac)
- Plusieurs librairies R
- Accès au web (une copie des pages/fichiers est fournie, quand même)

## Principes directeurs

Nous vivons dans un monde où les données (data) sont devenues une ressource très prisée

Beaucoup de données sont accessibles librement

Un modélisateur n'est pas *obligé* d'utiliser des données, mais quand des données sont disponibles, vous devriez au moins essayer de voir de quoi il retourne

Si vous voulez "avoir un impact" (pouvoir influencer une politique de santé publique), oubliez la stabilité globale!

1 Les données sont partout

2 Exemple – Graphiose de l'orme

# Données propriétaires versus données libres (open data)

## Données propriétaires

- souvent générées par des compagnies, gouvernements ou laboratoires de recherche
- quand elles sont disponibles, viennent avec multiples restrictions

## Open data

- souvent générées par les mêmes entités (compagnies, labos) mais *libérées* après une certaine période
- de plus en plus fréquent pour les gouvernements/entités publiques
- grande variété de licenses, donc attention
- grande variété de qualités, donc attention

## Initiatives Open Data

Mouvement récent (5-10 ans): governments (locaux ou plus haut) créent des portails où les données sont centralisées et publiées

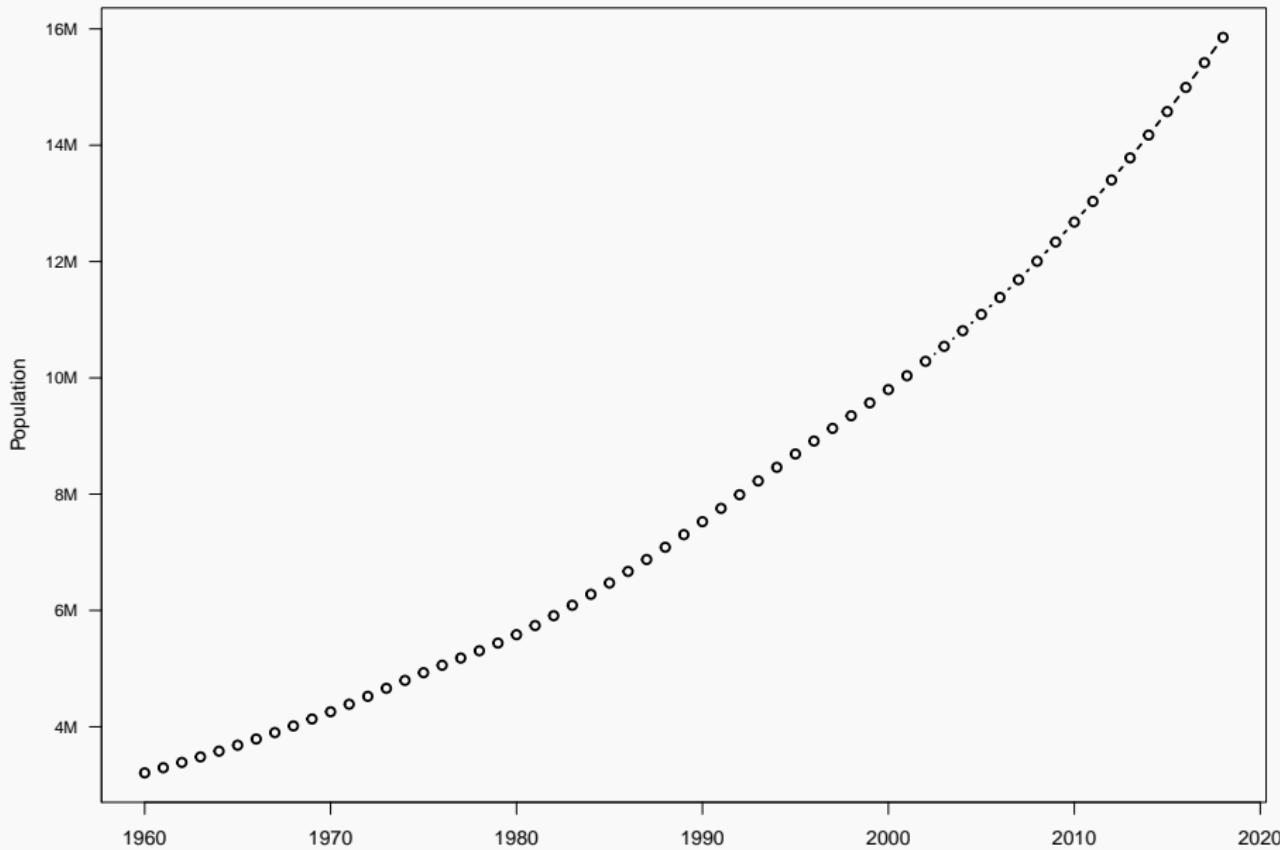
- <https://data.winnipeg.ca/>
- <https://open.alberta.ca/opendata>
- <https://open.canada.ca/en/open-data>
- <https://data.europa.eu/euodp/data/>
- <http://data.un.org/>
- <https://data.worldbank.org/>
- <https://www.who.int/gho/database/en/>

# Méthodes de récupération des données

- A la main
- En utilisant des programmes comme Engauge Digitizer ou g3data
- En utilisant des API
- En utilisant des processeurs de langage naturel (web scraping)
- En utilisant des packages (R, Python principalement)

## Exemple: la population du Sénégal

```
if (FALSE) {  
  pop_data_CTRY <- wb(country = c("SEN"), indicator = "SP.POP.TOTL",  
    mrv = 100)  
} else {  
  pop_data_CTRY = readRDS("pop_data_CTRY_downloaded.Rds")  
}  
y_range = range(pop_data_CTRY$value)  
y_axis <- make_y_axis(y_range)  
pdf(file = "pop_SEN.pdf", width = 11, height = 8.5)  
plot(pop_data_CTRY$date, pop_data_CTRY$value * y_axis$factor,  
  xlab = "Année", ylab = "Population", type = "b", lwd = 2,  
  yaxt = "n")  
axis(2, at = y_axis$ticks, labels = y_axis$labels, las = 1, cex.axis =  
dave <- dev.off()
```



1 Les données sont partout

2 Exemple – Graphiose de l'orme

## Graphiose de l'orme

- Maladie fongique de l'orme
- Causée par le champignon *Ophiostoma ulmi*
- Transmise par le scolyte de l'orme (*Scolytus scolytus*) (coléoptère)
- A décimé les forêts urbaines en amérique du nord

 Search[City of Winnipeg](#) [Open Data Catalogue](#) [Videos](#) [Developers](#) [Open Government Policy](#) [Open Data Policy](#)  [Sign In](#)

# City of Winnipeg Open Data Portal

## Open Capital Projects Dashboard

An interactive tool that reports on the performance of the City's open capital

## Open Budget

An interactive tool to inform citizens on the financial status of all open capital projects.

## Governance

Includes Hansards, dispositions, ward boundaries

## Finance

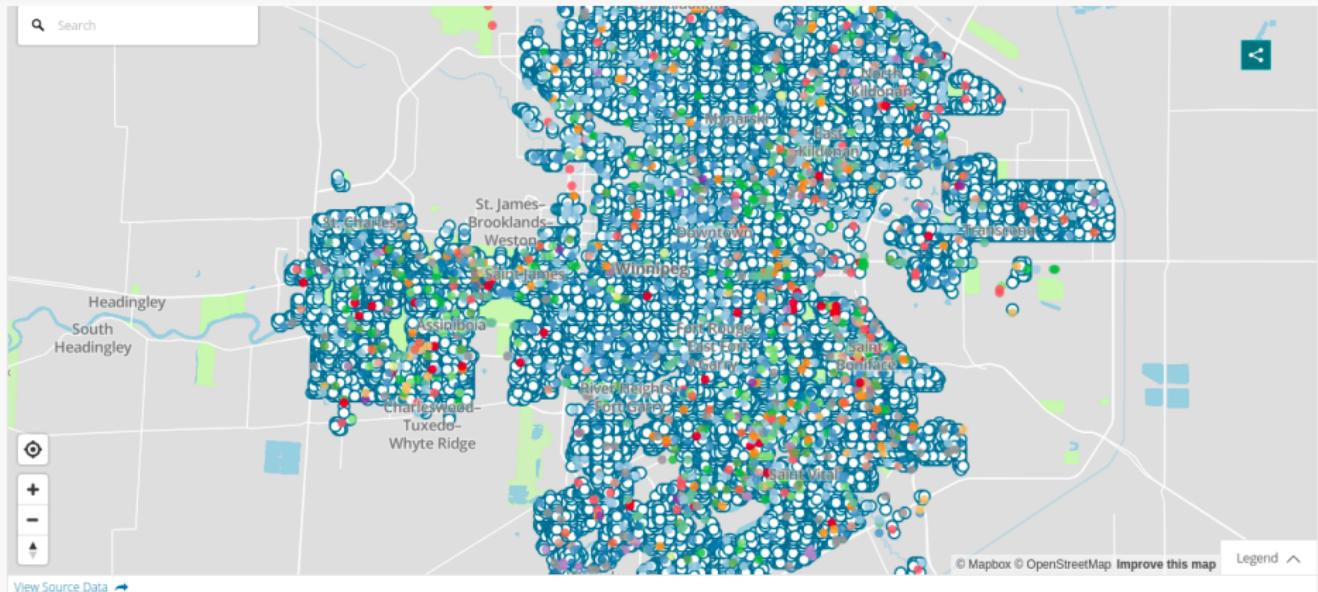
Includes capital and operating budgets, Council expenses



## City Services

## Transit API

## Maps



Tree ID	Botani...	Comm...	Electo...	Neigh...	Diamet...	Park	Locati...	Propre...	Street...	Street...	X	Y	DED T...	Locati...
1000	Ulmus dav...	Japanese el...	Old Kildon...	THE MAPLES	20	Not In Park	Boulevard	Public	Margate Rd	Massena Pl	631,154.17...	5,534,693....		(49.950179...
10000	Fraxinus p...	green ash	Daniel McL...	ST. MATTH...	46	Not In Park	Boulevard	Public	Ellice Av	St Matthe...	631,857.53...	5,528,241....		(49.892025...
100001	Fraxinus p...	green ash	St. Boniface	NIAKWA P...	39	Windsor P...	Park	Public	Cottonwoo...	Elizabeth Rd	636,953.38...	5,525,312....		(49.864546...
100004	Ouercus m...	bur oak	St. Boniface	NIAKWA P...	34	Windsor P...	Park	Public	Cottonwoo...	Elizabeth Rd	636,949.99...	5,525,307....		(49.864501...



Tree ID	Botani...	Comm...	Electo...	Neigh...	Diamet...	Park	Locati...	Prope...	Street...	Street...	X	Y	DED T...	Locati...
	https://data.winnipeg.ca/d/hfwk-jp4h	Old Kildon...	THE MAPLES		20	Not In Park	Boulevard	Public	Margate Rd	Massena Pl	631,154.17...	5,534,693....		(49.950179...

# Recuperation des données des arbres

```
allTrees = read.csv("https://data.winnipeg.ca/api/views/hfwk-jp4h/rows.
```

Voilà ce que cela retourne:

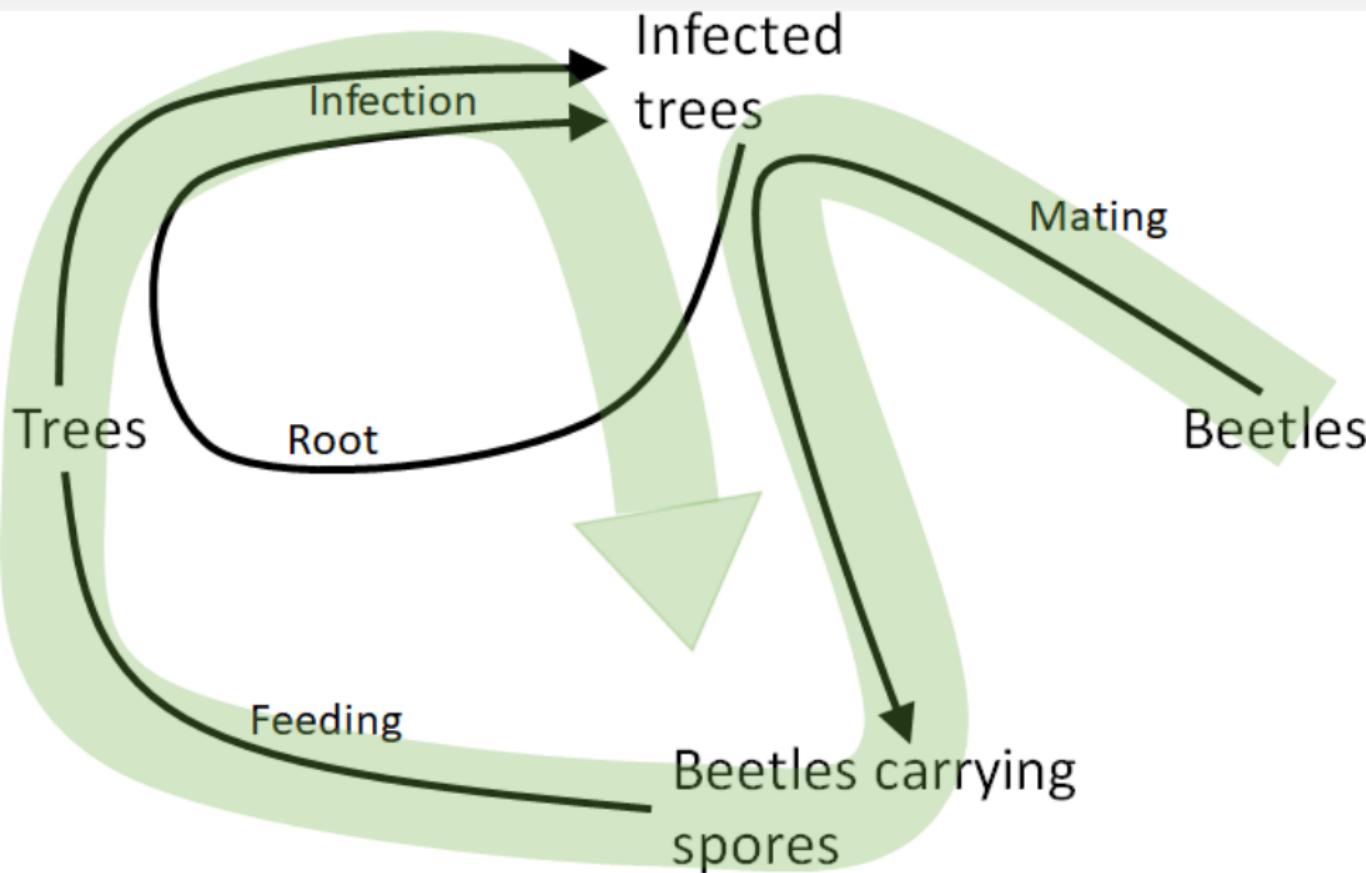
```
dim(allTrees)
```

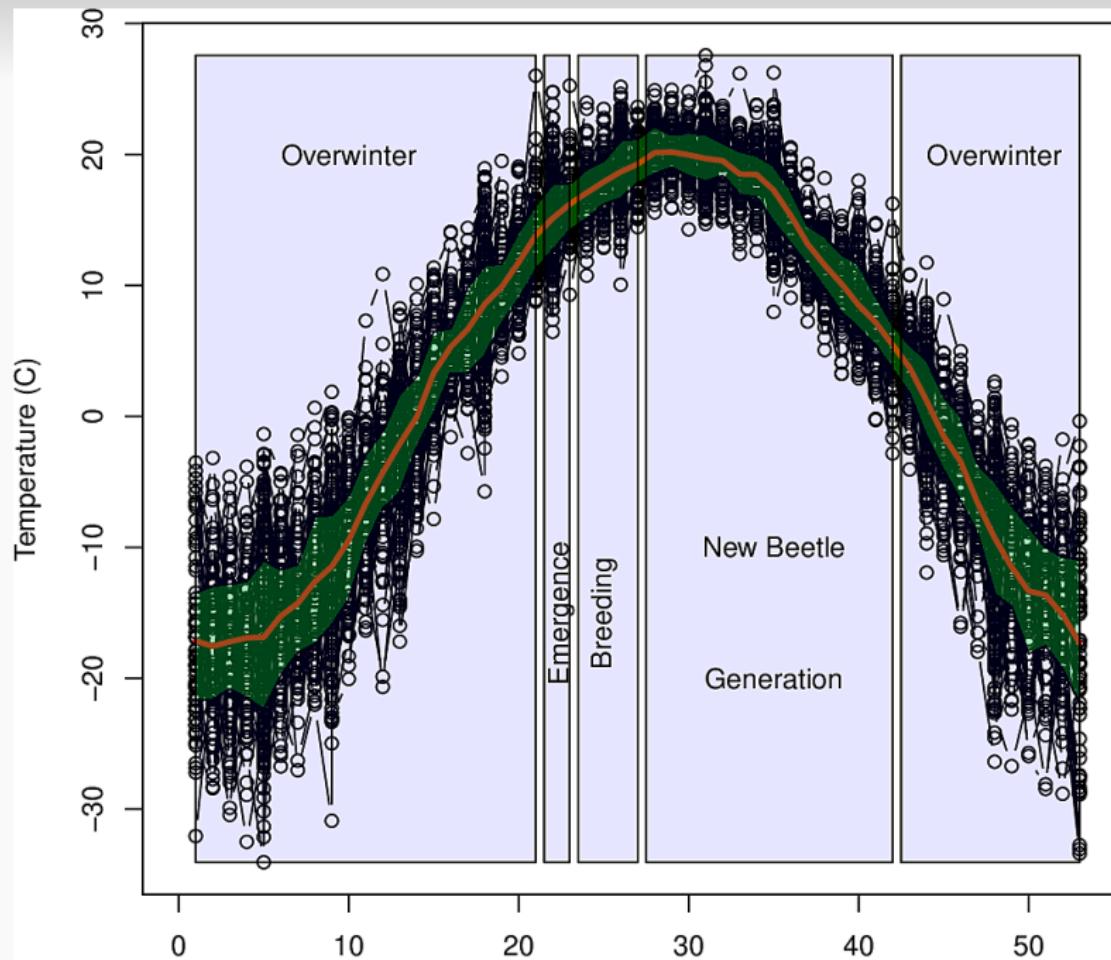
```
## [1] 291683      17
```

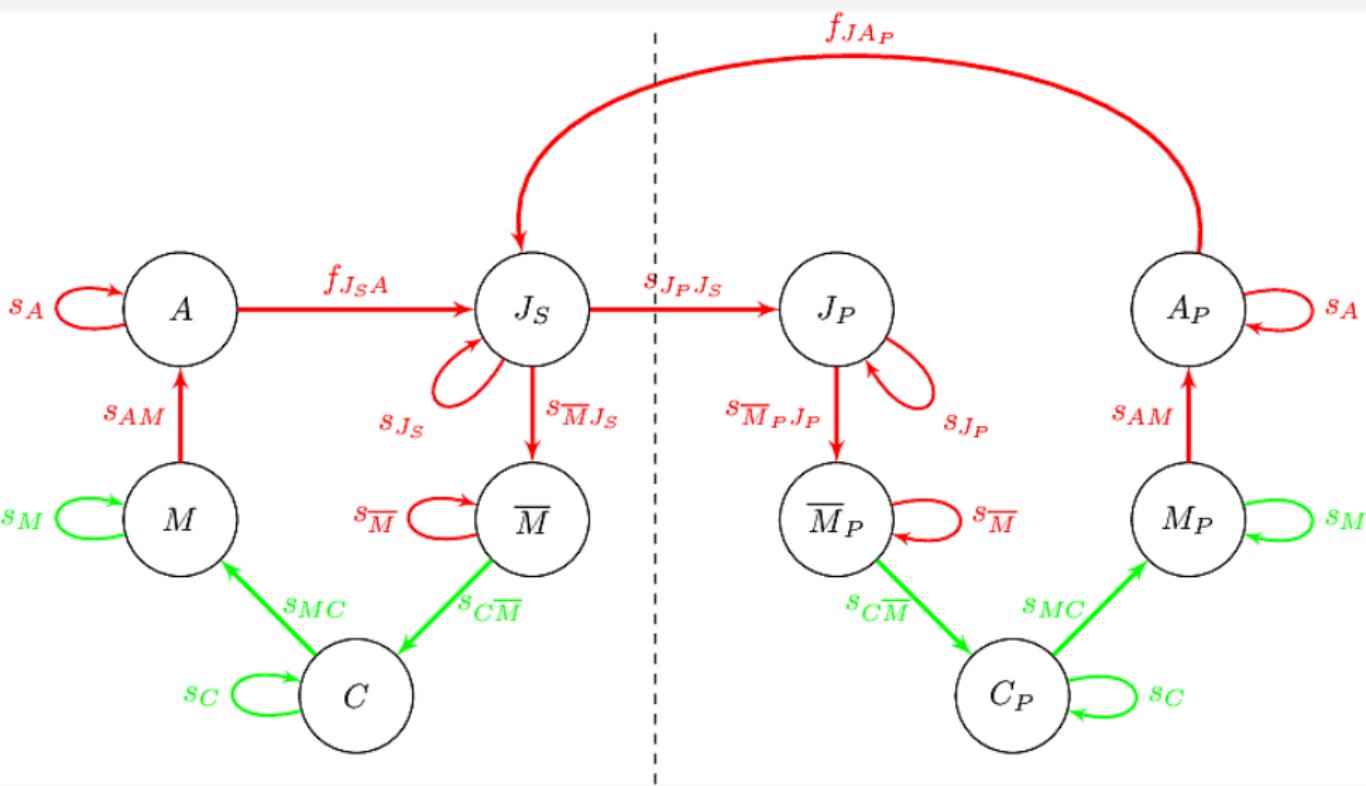
## On nettoie un peu

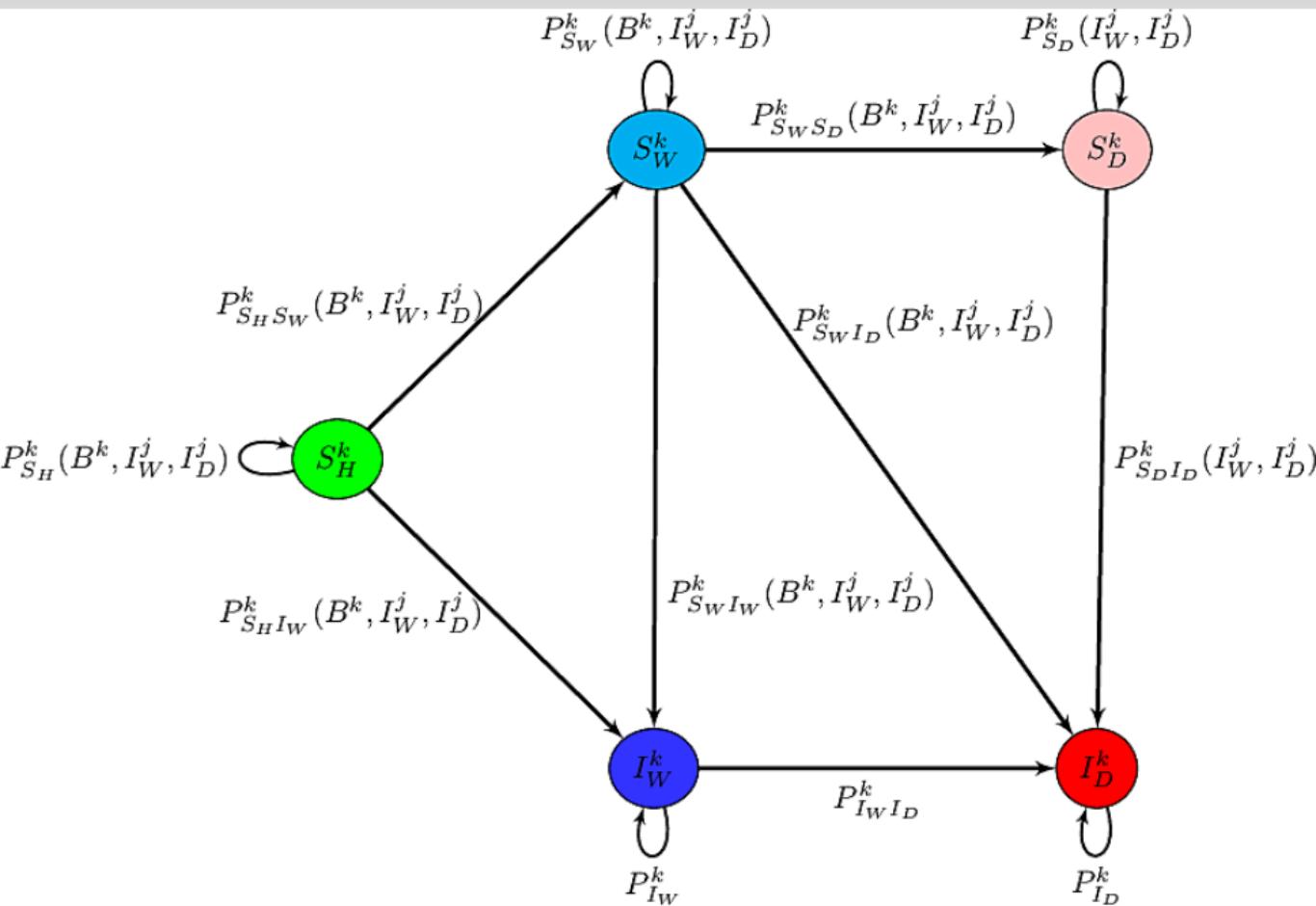
```
elms_idx = grep("American Elm", allTrees$Common.Name, ignore.case = TRUE)
elms = allTrees[elms_idx, ]
```

ce qui nous laisse 47307 ormes américains









## Calcul des interactions racinaires

(Pas en temps réel ici, on a besoin d'une machine assez grosse – environ 50GB de RAM)

- Si les racines d'un arbre infecté touchent les racines d'un arbre sain, le champignon est transmis
- Etendue système racinaire dépend de la hauteur de l'arbre (on a DBH)
- La façon dont les routes sont construites empêche les contacts racinaires d'un côté de la route à l'autre

## Distances entre tous les ormes

```
elms_xy = cbind(elms$X, elms$Y)
D = dist(elms_xy)
idx_D = which(D<50)
```

indices\_LT est une grosse ( $N(N - 1)/2 \times 2$ ) matrice avec les indices (*orig, dest*) des arbres dans les paires d'ormes, donc indices\_LT[idx\_D] sont les paires considérées

On garde un peu plus..

```
indices_LT_kept = as.data.frame(cbind(indices_LT[idx_D,],
                                         D[idx_D]))
colnames(indices_LT_kept) = c("i", "j", "dist")
```

## On crée des segments pour toutes les paires

```
tree_locs_orig = cbind(elms_latlon$lon[indices_LT_kept$i],  
                      elms_latlon$lat[indices_LT_kept$i])  
tree_locs_dest = cbind(elms_latlon$lon[indices_LT_kept$j],  
                      elms_latlon$lat[indices_LT_kept$j])  
tree_pairs = do.call(sf::st_sfc,  
                     lapply(  
                           1:nrow(tree_locs_orig),  
                           function(i){  
                             sf::st_linestring(  
                               matrix(  
                                 c(tree_locs_orig[i,],  
                                   tree_locs_dest[i,]),  
                                 ncol=2,  
                                 byrow=TRUE)  
                           )  
                         }  
                     ))
```

## Un peu de cartographie

```
library("tidyverse")
# Get bounding polygon for Winnipeg
bb_poly = osmdata::getbb(place_name = "winnipeg",
                         format_out = "polygon")
# Get roads
roads <- osmdata::opq(bbox = bb_poly) %>%
  osmdata::add_osm_feature(key = 'highway',
                           value = 'residential') %>%
  osmdata::osmdata_sf () %>%
  osmdata::trim_osmdata (bb_poly)
# Get rivers
rivers <- osmdata::opq(bbox = bb_poly) %>%
  osmdata::add_osm_feature(key = 'waterway',
                           value = "river") %>%
  osmdata::osmdata_sf () %>%
  osmdata::trim_osmdata (bb_poly)
```

## Et on termine tranquille

- On a les paires d'arbres potentiellement en contact
- On a les routes et les rivières de la ville (collection de segments de droite)
- Si il y a une intersection entre une paire et une route/rivière, on oublie cette paire

```
st_crs(tree_pairs) = sf::st_crs(roads$osm_lines$geometry)
iroads = sf::st_intersects(x = roads$osm_lines$geometry,
                           y = tree_pairs)
irivers = sf::st_intersects(x = rivers$osm_lines$geometry,
                           y = tree_pairs)
```

```
tree_pairs_roads_intersect = c()
for (i in 1:length(iroads)) {
  if (length(iroads[[i]])>0) {
    tree_pairs_roads_intersect = c(tree_pairs_roads_intersect,
                                    iroads[[i]])
  }
}
tree_pairs_roads_intersect = sort(tree_pairs_roads_intersect)
to_keep = 1:dim(tree_locs_orig)[1]
to_keep = setdiff(to_keep,tree_pairs_roads_intersect)
```

Les données sont partout

Exemple – Graphiose de l'orme



