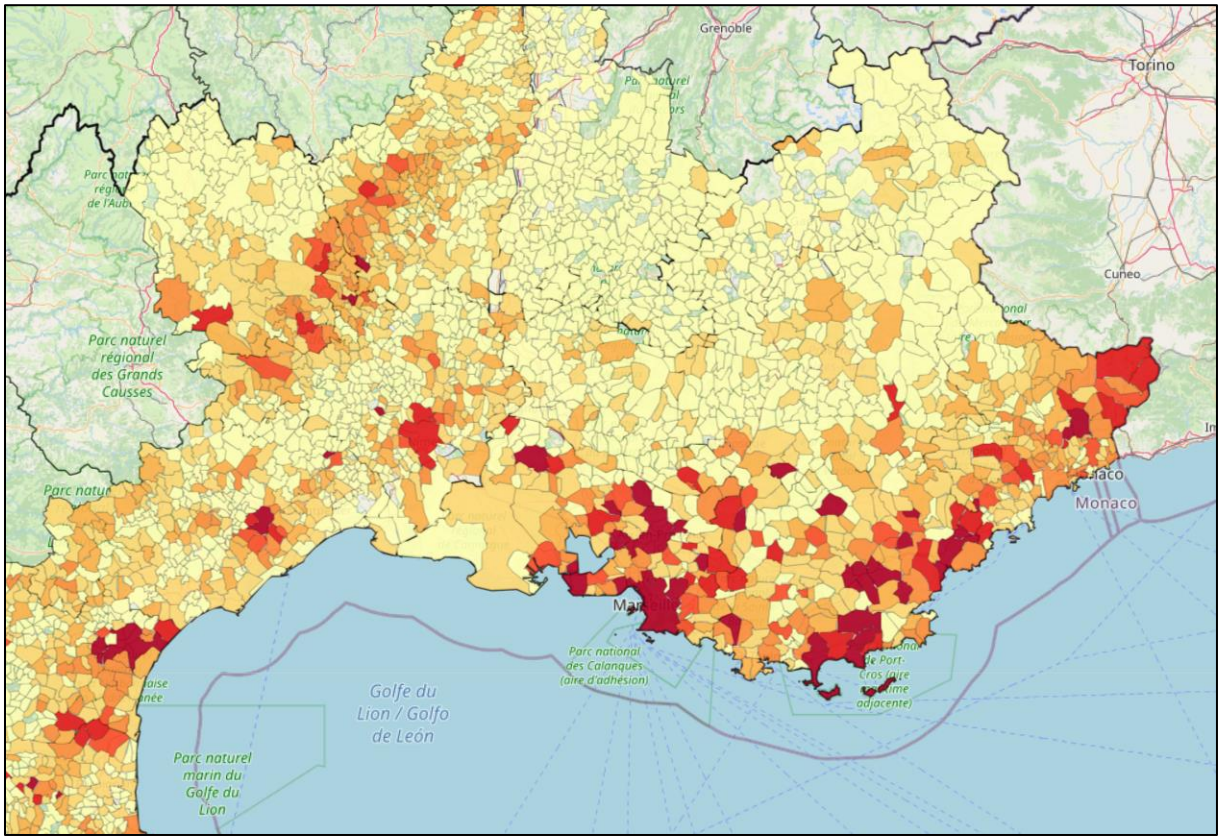
Suivi des incendies de forêts en bassin méditerranéen

Mise en forme des données de feux de la base prométhée grâce au langage de programmation R.

Géodatavisualisation (HGACU04)



1: Aperçu du la carte interactive (map.html) représentant la surface brulée (ha) selon les communes du bassin méditerranéen.

Master 2 : GAED parcours GMS

Faculté ALLSH d'Aix-Marseille Université

Problématique

Le bassin méditerranéen est une zone très touchée par les feux de forêt. En effet, pendant la période qui s'étend de 1973 à aujourd'hui cela représente quasiment 115 000 feux pour une surface brûlée de 932 426 ha (Prométhée, 2020). Comme nous pouvons le voir avec la Figure 2 et la Figure 3, le nombre de feux reporté chaque année ainsi que la surface brulée mesurée chaque année sont en diminution.

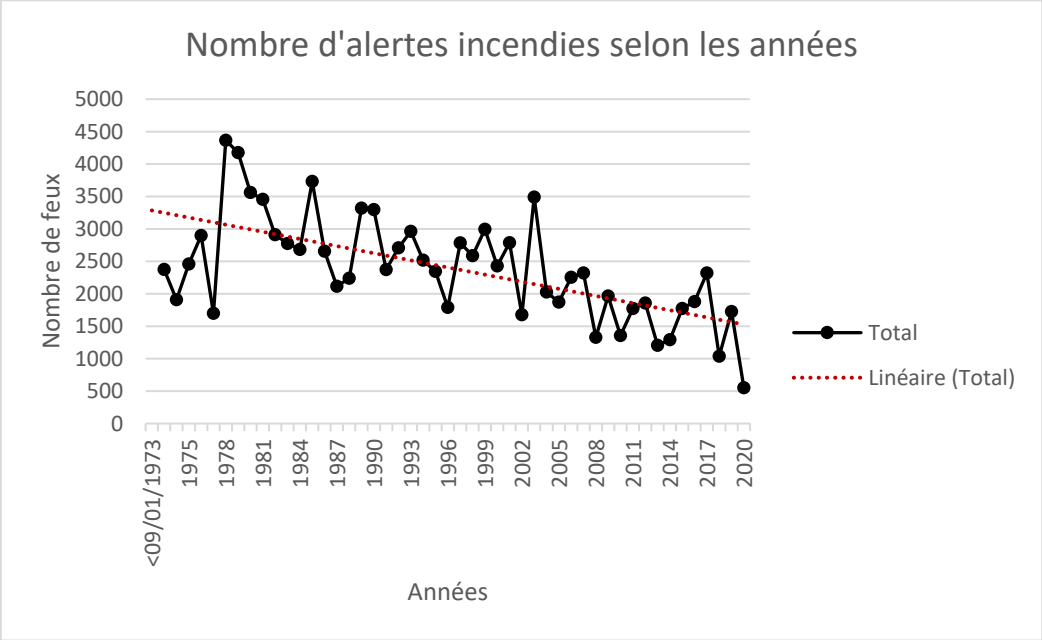


Figure 2: Nombre d'alertes incendies selon les années

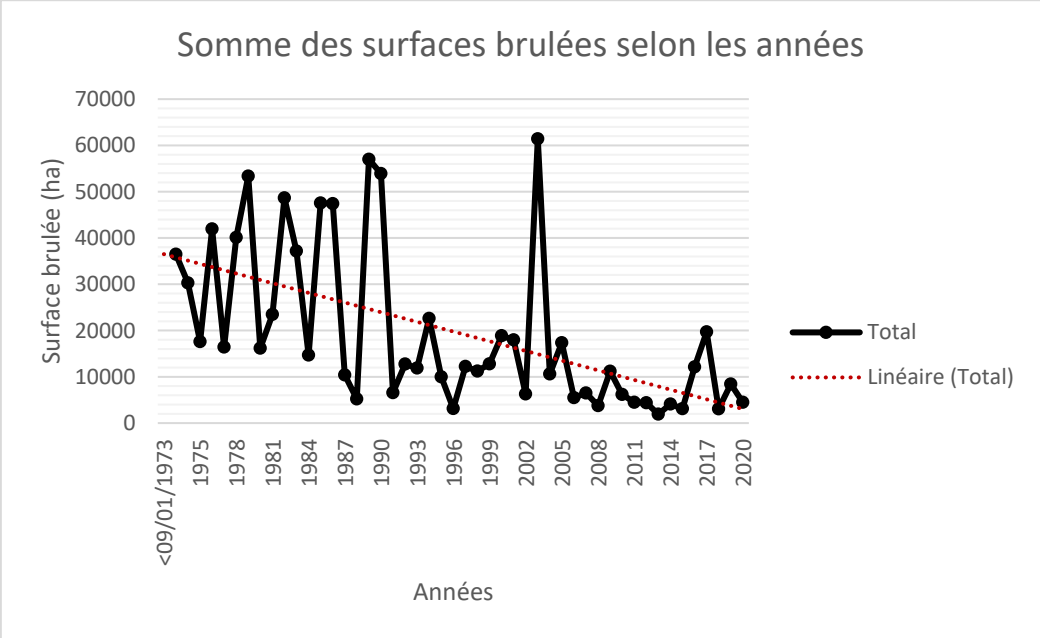


Figure 3: Somme des surfaces brulées selon les années

La problématique qui s'impose cependant est de savoir où est-ce que les feux se produisent et avec quelle intensité. En effet, ces informations sont nécessaires afin de répartir plus efficacement encore les moyens mis en place pour lutter contre les feux de forêts.

Pour cela je vais utiliser les données d'incendies de la base de données Prométhées que je vais mettre en forme puis cartographier selon les communes et le carroyage DFCI en utilisant R afin d'y ajouter une dimension dynamique.

Code

Pour chaque bout de code présenté ci-dessous, le détail précis est commenté par un # en début de ligne.

map.R

Tout d'abord, je charge les modules dont j'aurais besoin par la suite dans le script. Je le fais tout en haut du fichier car c'est plus clair et permet de détecter une quelconque erreur de versions avant de faire les calculs.

```
library(xts)
library(dplyr)
library(rgdal)
library(leaflet)
library(quantmod)
library(htmltools)
library(htmlwidgets)
library(RColorBrewer)
```

Par la suite, je change le *working directory* pour lire diverses données (csv de feux issus de la base de données prométhées, données vecteurs des régions, départements, communes, EPCI, et carroyage DFCI à 2 et 20km), que j'ai stocké dans le sous-dossier data.

```
# Je definit le path du wd comme etant le sous dossier data pour lire les
couches sig
setwd("./data/")

# Je lis les donnees brutes en csv de l'export de la base promethee
df_feux <- read.csv2("liste_incendies_ du_07_12_2020_formatte.csv")

# Tableaux croises dynamiques nombre, surface totale, surface max sur :
# Annee (graphe) commenter
# mois et heure (surface) commenter
# departements (graphe) commenter

# Je lis la couche vecteur des regions, departements, communes, epci
df_reg <- readOGR(dsn=getwd(), layer="REGION")
df_dep <- readOGR(dsn=getwd(), layer="DEPARTEMENT")
df_com <- readOGR(dsn=getwd(), layer="COMMUNE")
df_epci <- readOGR(dsn=getwd(), layer="EPCI")

# Je lis les couches vecteurs des carroyages DFCI
df_dfci2 <- readOGR(dsn=getwd(), layer="DFCI2")
df_dfci20 <- readOGR(dsn=getwd(), layer="DFCI20")

# Je redefinit le path du wd un dossier au dessus dans la racine pour
retourner dans le dossier
# principal du projet.
```

```
setwd("..")
```

Une fois les données chargées, je fais des regroupements selon les communes ou les carreaux DFCI puis des jointures avec les shapefiles correspond pour la mise en carte.

```
# Communes
# Je fais un group by pour regrouper les donnees selon le champ
code_INSEE puis un summarise pour y associer
# les donnees de surface brulee ainsi que de nombre de feux
df_feux_gb_com <- df_feux %>% group_by(code_INSEE) %>%
summarize(surface_ha=sum(surface_ha), nbr_feux=n())

# Je fait une jointure du shapefile des communes et des donnees
promethees de feux que j'ai regroupe
# juste au dessus en utilisant les champs du code insee
df_feux_com = df_com %>%
  merge(
    x=df_com,
    y=df_feux_gb_com,
    by.x="INSEE_COM",
    by.y="code_INSEE"
  )

# Je supprime le dataframe df_feux_gb_com car je n'en ai plus besoin (les
donnees sont integrees dans le
# dataframe df_feux_com)
rm(df_feux_gb_com)

# Je supprime les communes qui ne comportent pas de feux (celles qui
n'ont pas de valeur dans le champ nbr_feux)
#df_feux_com <- subset(df_feux_com, df_feux_com$surf_parcourue_m2 != 0)
df_feux_com = subset(df_feux_com, df_feux_com@data$nbr_feux != "")

# DFCI2

# Je fais un group by pour regrouper les donnees selon le champ code_DFCI
puis un summarise pour y associer
# les donnees de surface brulee ainsi que de nombre de feux
df_feux_gb_dfci2 <- df_feux %>% group_by(code_DFCI) %>%
summarize(surface_ha=sum(surface_ha), nbr_feux=n())

# Je fait une jointure du shapefile du carroyage DFCI a 2km et des
donnees promethees de feux que j'ai
# regroupe juste au dessus en utilisant les champs du code DFCI
df_feux_dfci2 = df_dfci2 %>%
  merge(
    x=df_dfci2,
    y=df_feux_gb_dfci2,
    by.x="NOM",
    by.y="code_DFCI"
  )

# Je supprime le dataframe df_feux_gb_dfci2 car je n'en ai plus besoin
(les donnees sont integrees dans le
# dataframe df_feux_dfci2)
rm(df_feux_gb_dfci2)

# Je supprime les carreaux DFCI qui ne comportent pas de feux (ceux qui
n'ont pas de valeur dans le champ nbr_feux)
df_feux_dfci2 = subset(df_feux_dfci2, df_feux_dfci2@data$nbr_feux != "")
```

```
# DFCI20
# Je fais un group by pour regrouper les donnees selon le champ code_DFCI
puis un summarise pour y associer
# les donnees de surface brulee ainsi que de nombre de feux
df_feux_gb_dfci20 <- df_feux %>% group_by(code_DFCI) %>%
summarize(surface_ha=sum(surface_ha), nbr_feux=n())

# Je fait une jointure du shapefile du carroyage DFCI a 20km et des
donnees promethees de feux que j'ai
# regroupe juste au dessus en utilisant les champs du code DFCI
df_feux_dfci20 = ddf_dfci20 %>%
  merge(
    x=df_dfci20,
    y=df_feux_gb_dfci20,
    by.x="NOM",
    by.y="code_DFCI"
  )

# Je supprime le dataframe df_feux_gb_dfci20 car je n'en ai plus besoin
(les donnees sont integrees dans le
# dataframe df_feux_dfci20)
rm(df_feux_gb_dfci20)

# Je supprime les carreaux DFCI qui ne comportent pas de feux (ceux qui
n'ont pas de valeur dans le champ nbr_feux)
df_feux_dfci20 = subset(df_feux_dfci20, df_feux_dfci20@data$nbr_feux !=
"")
```

Une fois que les données sont jointes et prêtes à être mises en carte, je créé la palette de couleur qui constituera la symbologie graphique de la carte.

```
# Je définit les écarts de valeurs pour la légende et la coloration
bins <- c(0, 100, 250, 500, 1000, 2500, Inf)

# J'associe les écarts de valeurs avec une palette de couleur et un champ
# La symbologie graphique est créé
palette_feux_com <- colorBin("YlOrRd",
domain=df_feux_com@data$surface_ha, bins=bins)
palette_feux_dfci2 <- colorBin("YlOrRd",
domain=df_feux_dfci2@data$surface_ha, bins=bins)
```

Dans un second temps, il faut que je définisse le format dans lequel les données seront affichées dans les popups. Ici, je définit les contenus des popups pour les deux couches car je veux qu'elles s'affichent de manière dynamique selon la couche que l'on affiche avec le *LayersControl*.

```
# Je définit le contenu des popups pour la couche des feux selon la
commune
popup_com <- paste("Commune: ", df_feux_com@data$NOM_COM_M, "<br/>",
                   "Surface brulée: ",
round(df_feux_com@data$surface_ha), "ha",
                   sep="") %>% lapply(htmltools::HTML)

# Je définit le connu des popups pour la couhe des feux selon le
carroyage dfci
popup_dfci2 <- paste("Carreau DFCI: ", df_feux_dfci2@data$NOM, "<br/>",
```

```
                    "Surface brulée: ",
round(df_feux_dfci2@data$surface_ha), "ha",
                    sep="") %>% lapply(htmltools::HTML)
```

Enfin, une fois que tous les éléments minimums sont générés, je peux créer le canevas de la carte. Cela passe tout d'abord par le fait de définir le fond de carte et la zone que l'on voit par défaut. Par la suite, j'ajoute les polygones administratifs (couches vecteurs), à la carte qui ne contiennent pas de données de feux (régions, départements et epci). J'ajoute ces couches car elles permettent de mieux se repérer qu'avec le fond OSM uniquement.

```
# Je créé ma carte leaflet de base avec
map <- leaflet() %>%

  # Localisation de base de la carte lorsqu'elle est initialisée
  setView(5, 45, 6) %>%

  # Ca sert à rien mais c'est pas pour la place que ça prend
  #addMiniMap(toggleDisplay=TRUE)

  # Ajout du fond de carte: group correspond au nom que l'on veut donner
au fond de carte
  # (C'est ce nom que l'on devrait encapsuler dans un groupe du
LayersControl et qui devrait apparaître dans
  # la légende). Moi je met que OSm car ça fait très bien le travail
  addTiles(group="OSM (default)") %>%

  # Ajout des polygones des régions, départements, epci, communes et
définition pour chacun
  # de leur style graphique (sert de fond de carte donc n'utilise pas les
données de feux
  # pour faire des graduations)
  addPolygons(data=df_reg, fill=FALSE, weight=2, color="#000",
group="Régions") %>%
  addPolygons(data=df_dep, fill=FALSE, weight=1, color="#000",
group="Départements") %>%
  addPolygons(data=df_epci, fill=FALSE, weight=0.5, color="#000",
group="EPCI") %>%
```

Puis, toujours dans la création de la carte leaflet, j'ajoute ensuite de la même manière les polygones contenant les informations de feux cette fois. A noter que j'utilise l'argument *fillColor* pour déterminer la symbologie à utiliser et l'argument *label* pour déterminer le format de popup que je veux utiliser. De la même manière, je définis la légende après le processus d'import des couches vecteurs avec la fonction *leaflet ::addLegeng()*.

```
  # AJout des données de feux
  # Ajout des données de feux selon les communes
  addPolygons(
    data=df_feux_com,
    fillColor=palette_feux_com(df_feux_com@data$surface_ha),
    fillOpacity = 0.9,
    color="black",
    group="Communes",
    weight=0.3,
    label=popup_com) %>%

  # Ajout des données DFCI à 2km
  addPolygons(
    data=df_feux_dfci2,
```

```
    fillColor=palette_feux_dfci2(df_feux_dfci2@data$surface_ha),
    fillOpacity = 0.9,
    color="black",
    group="DFCI 2km",
    weight=0.3,
    label=popup_dfci2) %>%

  # Ajout de la légende
  # raise Error in get(".xts_chob", .plotxtsEnv) : objet '.xts_chob'
introuvable
  # écrire leaflet::addLegend au lieu de %>% addLegend() à l'air de
régler le problèm

  leaflet::addLegend(data=df_feux_com,
                     pal=palette_feux_com,
                     values=df_feux_com@data$surface_ha,
                     title="Surface brulée (ha)",
                     position="bottomleft",
                     group="Communes") %>%

  leaflet::addLegend(data=df_feux_dfci2,
                     pal=palette_feux_dfci2,
                     values=df_feux_dfci2@data$surface_ha,
                     title="Surface brulée (ha)",
                     position="bottomleft",
                     group="DFCI 2km") %>%
```

Dès que les couches et popups importées, je met en place le système de gestion de visibilité des couches et cache les plus lourds afin d'accélérer l'affichage de la carte à l'ouverture. Puis, j'ajoute une échelle graphique automatique. Enfin, lorsque la carte est terminée, je la sauvegarde avec la fonction *saveWidget()*.

```
  # Ajout du menu de control des couches et regroupement des couches par
groupes de control.
  addLayersControl(
    overlayGroups=c("Régions", "Départements", "EPCI", "Communes", "DFCI
2km"),
    options=layersControlOptions(collapsed=TRUE)) %>%

  # Je définit quelles couches sont cachées par défaut
  # Ca aide à ce que la carte charge plus vite.
  hideGroup("EPCI") %>%
  hideGroup("Communes") %>%
  hideGroup("DFCI 2km") %>%

  # Ajout tout simple de la barre d'échelle
  addScaleBar(position="bottomleft")

# Créé litéralement la carte en executant la fonction leaflet derrière
# C'est là que je génère le rendu de la carte dans le viewer en appelant
la fonction map
# (je préfère la sauvegarder en html directement à la fin du script)

# Sauvegarde map (la cartographie) vers le fichier map.html dans le wd
par défaut
# selfcontained=FALSE permet d'enregistrer (sinon ça crashait à cause du
manque de RAM
#(probablement car le html était trop lourd))
```

```
saveWidget(widget=map,
           file="map.html",
           selfcontained = FALSE)
```

plot.R

De la même manière que pour le script qui génère la carte, je commencer par importer les modules qui me serviront par la suite puis je lis les données de feux du fichier csv de la base de données prométhée.

```
library(dplyr)
library(plotly)
library(htmlwidgets)

# Je definit le path du wd comme etant le sous dossier data pour lire les
couches sig
setwd("./data/")

# Je lis les donnees brutes en csv de l'export de la base promethee
df_feux <- read.csv2("liste_incendies_ du_07_12_2020_formatte.csv")

# Je redefinit le path du wd un dossier au dessus dans la racine pour
retourner dans le dossier
# principal du projet.
setwd("..")
```

Par la suite, je regroupe les données grâce à la fonction *group_by()* selon le champ années puis je récupère ces données avec la fonction *summarize*.

```
# Je fais un group by pour regrouper les donnees selon un champ puis un
summarise pour y associer les donnees
df_feux_gb_annee_surfha_nbr <- df_feux %>% group_by(annee) %>%
summarize(surface_ha_max=round(max(surface_ha)),
sum_surface_ha=round(sum(surface_ha)), nbr=n())
```

Une fois les données mises en forme, j'utilises la fonction *plot_ly()* pour en faire un graphique à partir de deux champs x et y. Puis avec la fonction *add_trace()*, j'ajoute de nouvelles données en y. Les arguments nam

```
# C'est la que je met en forme le graph
# https://plotly.com/r/line-charts/
feu_par_ans <- plot_ly(
  data=df_feux_gb_annee_surfha_nbr,
  x=df_feux_gb_annee_surfha_nbr$annee,
  y=df_feux_gb_annee_surfha_nbr$sum_surface_ha,
  name="Surface totale brulee",
  type="scatter",
  mode="lines+markers",
  color=I('black')
) %>%
  add_trace(y=df_feux_gb_annee_surfha_nbr$surface_ha_max,
            name="Surface maximum brulee",
            type="scatter",
            mode="lines+markers",
            color=I("red")) %>%
  add_trace(y=df_feux_gb_annee_surfha_nbr$nbr,
            name="Nombre d'incendies",
            type="scatter",
            mode="lines+markers",
            color=I("blue"))
```

Toujours lors de la création du graphique mais seulement après avoir importé les données, je définis le fait que le graphique possède un slider pour affiner les dates.

```
# Je met en forme le titre du graph ainsi que les titres des axes
feu_par_ans <- feu_par_ans %>% layout(title = "Surface totale et maximum
brulee (ha) et
                                      nombre d'incendies selon les annees",
                                        # Met le mode "Compare data on
hover" actif par defaut.
                                        hovermode='compare',
                                        legend=list(x=0.75,
                                                    y=0.9),
                                        xaxis=list(title="Annees",
                                                   rangeslider=list(
                                                     type="date")),
                                        yaxis=list(visible=FALSE)) %>%
  config(displayModeBar = FALSE)  # Cache les commandes du graph par defaut.
```

Enfin, lorsque la création du graphique est terminée, je l'enregistre dans un fichier HTML puis je supprime les variables qui contenaient mes données pour libérer de l'espace de mémoire.

A noter que les paramètres passés dans la fonction *saveWidget()* sont différents de ceux utilisés pour sauvegarder le résultats de ma carte. En effet, sans l'argument *selfcontained = FALSE* il s'avère que *saveWidget()* embarque les données dans le fichier HTML. Pour de petits fichiers (taille sur le disque), cela marche très bien mais pour de plus gros fichiers comme la carte qui possède aussi un fond de carte, c'est beaucoup trop lourd et ça faisait planter Rstudio.

```
# C'est la que je genere le rendu du graph dans le viewer en appelant la
fonction graph
# (je prefere le sauvegarder en html directement)
#feu_par_ans
# Sauvegarde graph (le graphique 2D) vers le fichier feu_par_an.html dans
le wd par defaut
# (car je ne l'ai pas redetermine)
saveWidget(feu_par_ans, file="feu_par_an.html")

# Maintenant que c'est enregistre je peux supprimer toutes les variables
qui m'ont permise de generer le graph
rm(df_feux_gb_annee_surfha_nbr)
rm(feu_par_ans)
```

Par la suite, je fais exactement le même processus mais pour mettre en graphique les données de surface et de nombre selon les départements. Seule différence notable, ce second graphique est un histogramme selon les départements et non une courbe, représenté selon les années. Du coup, l'un possède un slider pour affiner les années et l'autre non.

```
# Je fais un group by pour regrouper les donnees selon un champ puis un
summarise pour y associer les donnees
df_feux_gb_dep_surfha_nbr <- df_feux %>% group_by(dep) %>%
summarize(surface_ha_max=round(max(surface_ha)),
sum_surface_ha=round(sum(surface_ha)), nbr=n())

# C'est la que je met en forme le graph
# https://plotly.com/r/line-charts/
feu_par_dep <- plot_ly(
  data=df_feux_gb_dep_surfha_nbr,
  x=df_feux_gb_dep_surfha_nbr$dep,
  y=df_feux_gb_dep_surfha_nbr$nbr,
```

```
  name="Nombre d'incendies",
  type="scatter",
  color=I("blue")
) %>%
  add_trace(y=df_feux_gb_dep_surfha_nbr$sum_surface_ha,
            name="Surface totale brulee",
            type="bar",
            color=I('black')) %>%
  add_trace(y=df_feux_gb_dep_surfha_nbr$surface_ha_max,
            name="Surface maximum brulee",
            type="bar",
            color=I("red"))

# Je met en forme le titre du graph ainsi que les titres des axes
feu_par_dep <- feu_par_dep %>% layout(title = "Surface totale et maximum
brulee (ha) et
                                      nombre d'incendies selon les
departements",
                                        # Met le mode "Compare data on
hover" actif par defaut.
                                        legend=list(x=0,
                                                    y=0.9),
                                        xaxis=list(title="Departements"),
                                        yaxis=list(visible=FALSE),
                                        barmode='overlay',
                                        bargap=0.15) %>%
  config(displayModeBar = FALSE)  # Cache les commandes du graph par
defaut.

# C'est la que je genere le rendu du graph dans le viewer en appelant la
fonction graph
# (je prefere le sauvegarder en html directement)
#feu_par_dep

# Sauvegarde graph (le graphique 2D) vers le fichier feu_par_an.html dans
le wd par defaut
# (car je ne l'ai pas redetermine)
saveWidget(feu_par_dep, file="feu_par_dep.html")

# Maintenant que c'est enregistre je peux supprimer toutes les variables
qui m'ont permise de generer le graph
rm(df_feux_gb_dep_surfha_nbr)
rm(feu_par_dep)
```

Résultats

Les résultats sont visibles dans les fichiers :

- feu_par_dep.html
- feu_par_an.html
- map.html

Ci-dessous, des aperçus des fichiers :

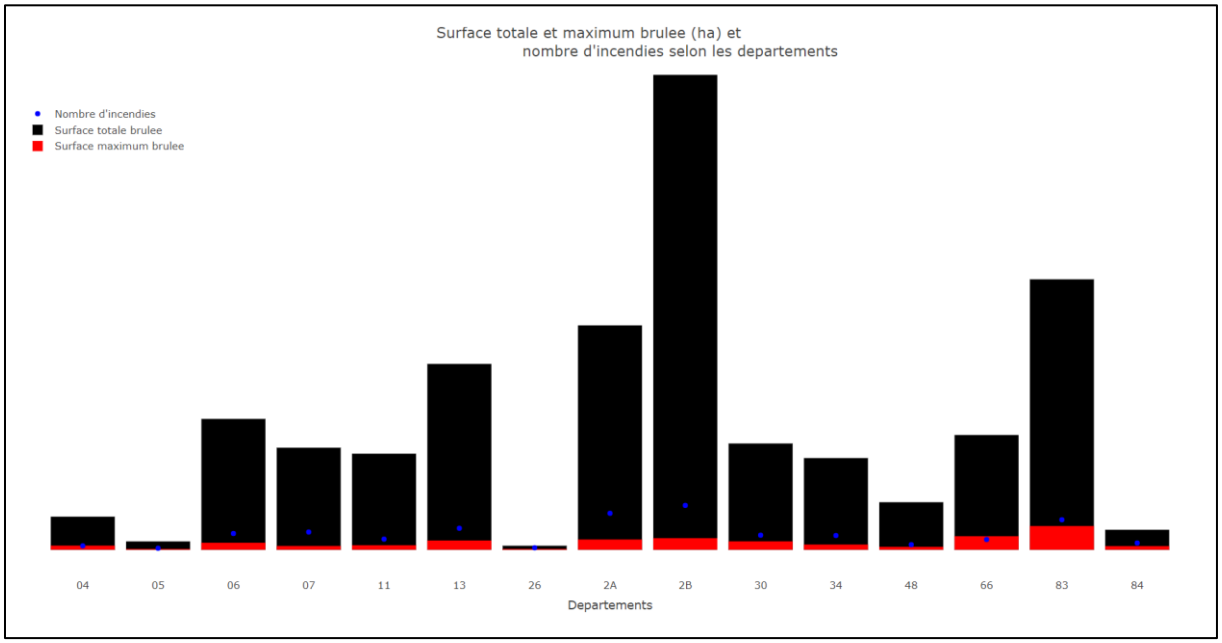


Figure 4: Surface totale et surface maximum brulée (ha) et nombre d'incendies selon les départements.

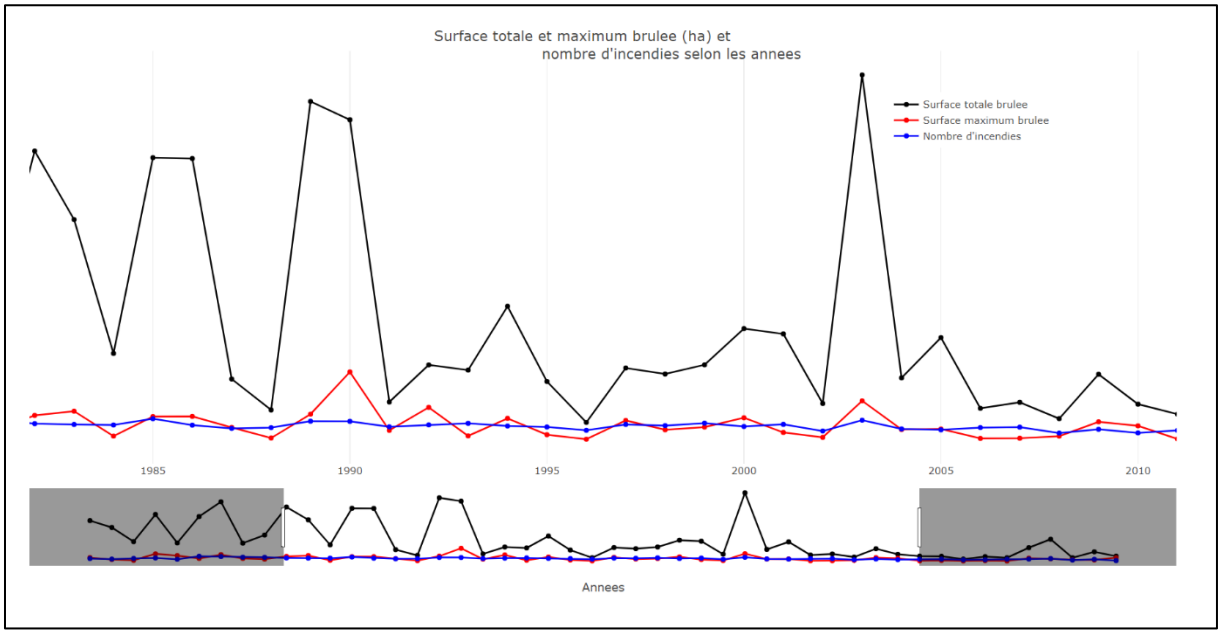


Figure 5: Surface totale et surface maximum brulée (ha) et nombre d'incendies selon les années.

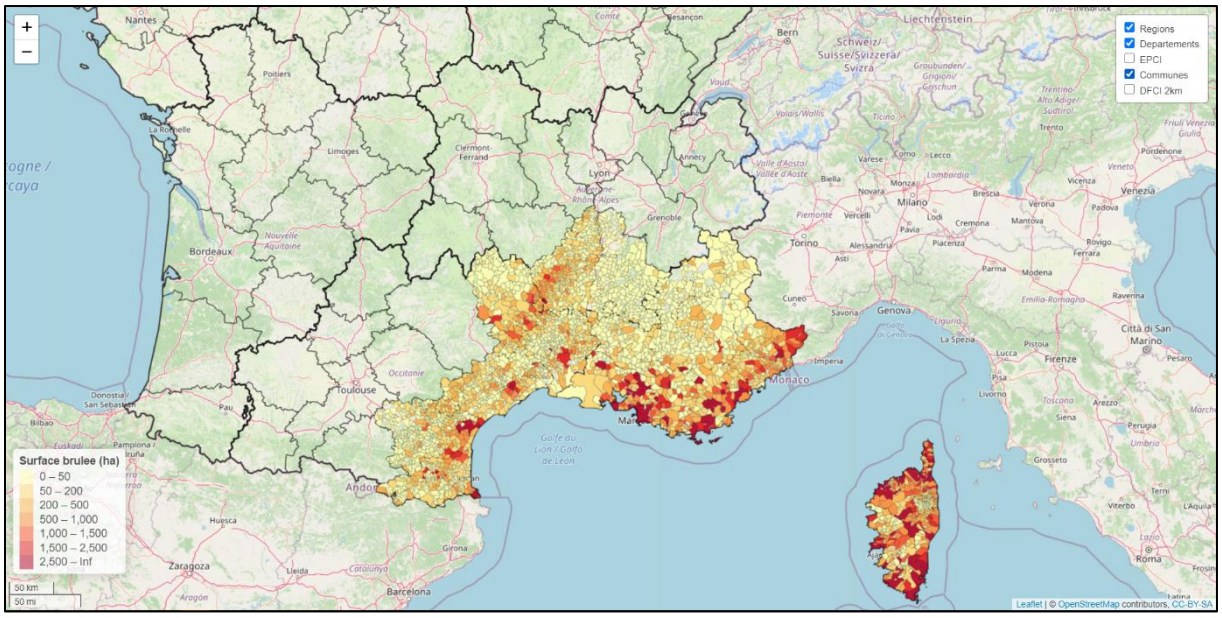


Figure 6: Surface brulée (ha)

Conclusion

Pour conclure, nous pouvons voir grâce à la cartographie dynamique (Figure 6), que dans le bassin méditerranéen, les feux sont le plus récurrents en littoral que dans les terres. De plus, grâce au graphique interactif (Figure 4), qui répertorie les feux selon les départements, nous pouvons voir que ce sont les Bouches du Rhône (13), le Var (83), les Alpes Maritimes (06) et la Corse (2A et 2B) qui sont le plus durement touchés. A l'inverse, nous pouvons aussi voir que des départements comme les Hautes-Alpes (05) et la Drôme (26) sont plus épargnés par les incendies.

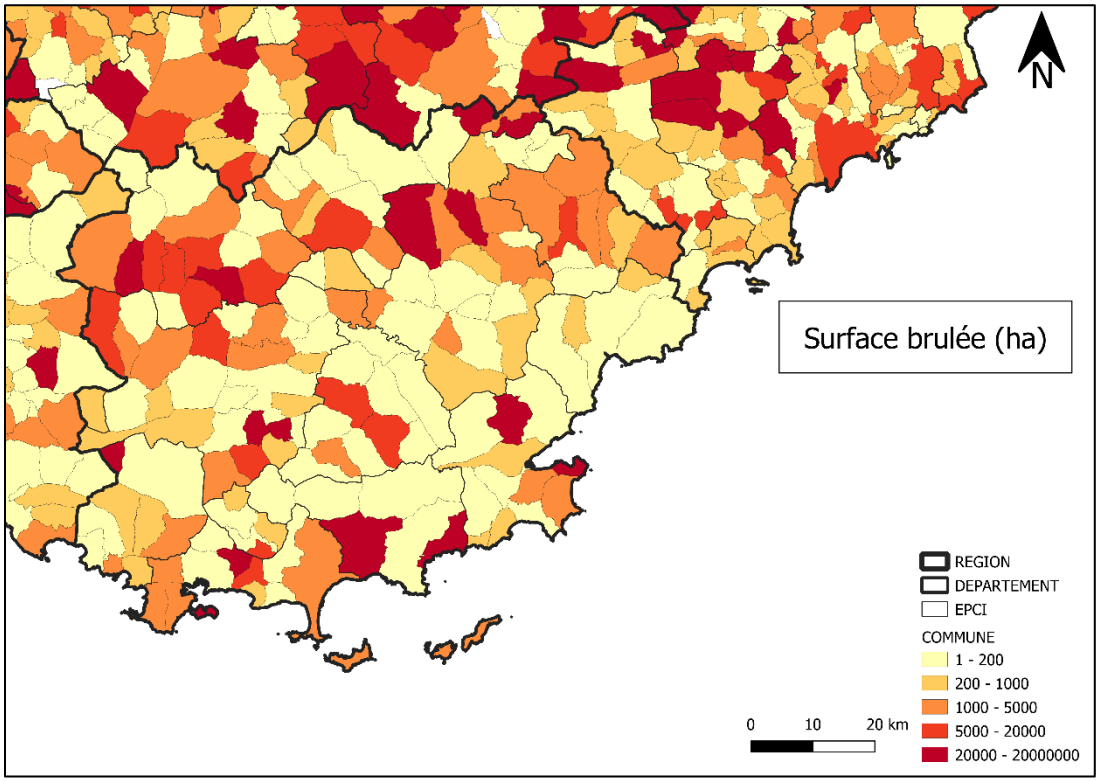
De plus, si l'on regarde la surface brulée moyenne (ha) plutôt que le nombre d'incendies répertoriés, nous pouvons observer une dynamique différente. Les départements les plus touchés selon la surface brulée (13, 83, 06, 2A et 2B), ne sont pas obligatoirement les plus les plus touchés selon le nombre d'incendies. En effet, mis à part les départements de la Corse (2A et 2B), nous pouvons voir grâce au Tableau 1 que des départements comme le 11, 30, 66 ou 84 répertorient une grande quantité de feux malgré une faible surface brulée.

Le calcul de la surface brulée en moyenne permet de bien montrer ces différences. Les départements du 11, 2B, 48, 66 et 83 sont ceux qui apparaissent comme ayant les feux les plus violents et le 05, le 07 et le 26 comme ceux ayant les feux les moins violents.

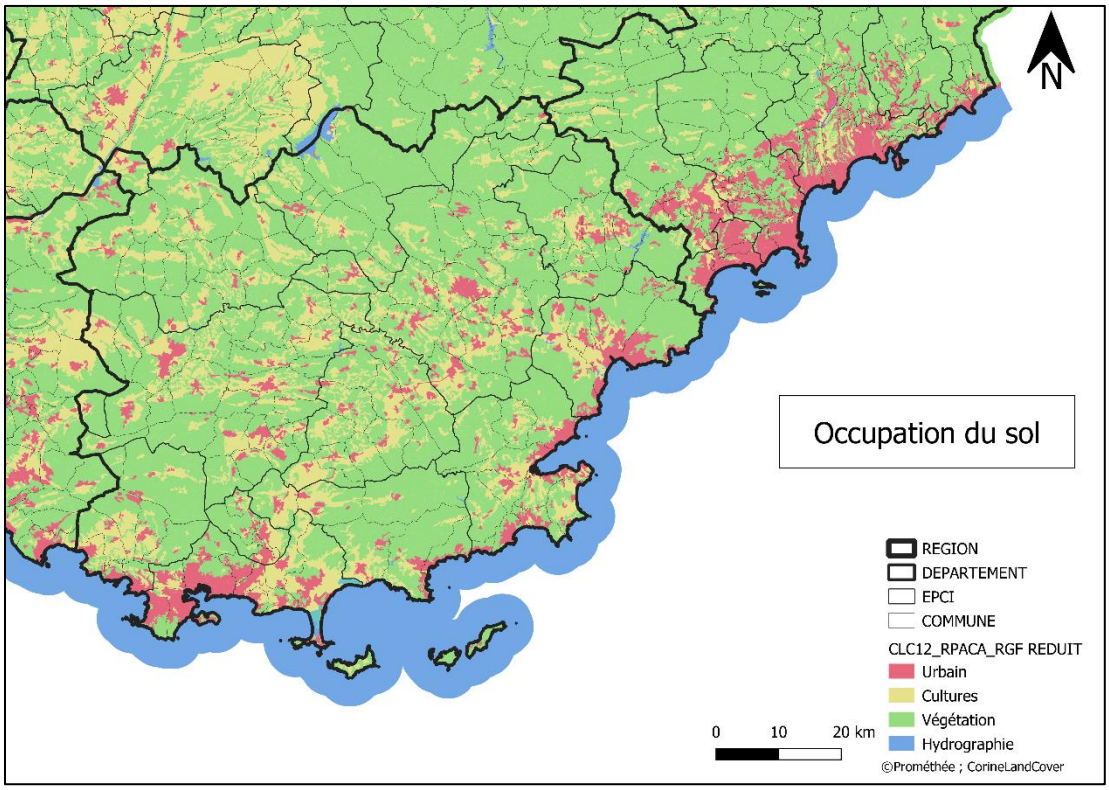
Département	04	05	06	07	11	13	26	2A	2B	30	34	48	66	83	84
Surf brulée (ha)	16134	4053	63872	49808	46891	90722	1938	109543	231863	51868	44755	23197	56014	132030	9739
Nbr feux	1903	727	7921	8638	5174	10467	945	17791	21645	7099	6936	2471	4981	14630	3284
Surf brulée (moy ha)	8.47	5.57	8.06	5.76	9.06	8.66	2.05	6.15	10.71	7.30	6.45	9.38	11.24	9.02	2.96

Tableau 1: Surface brulée et nombre d'incendies selon les départements

Enfin, la comparaison de bases de données géographiques d'occupation du sol telle que le CorineLandCover avec les données de surface brulée et de nombre d'incendies selon les communes ou le carroyage DFCI nous permettent aussi de connaître la nature des zones les plus touchées.



Carte 1: Surface brulée (ha)



Carte 2: Occupation du sol du département du var

D’après la Carte 1 et la Carte 2, nous pouvons observer que les zones les plus touchées par les incendies sont d’ailleurs les zones mixtes de cultures agricoles et de végétation (forêt, prairies, etc)

Bibliographie

Prométhée. (2020). Récupéré sur Prométhée: https://www.promethee.com/default/incendies