Visualisation graphique

Fabien Haury

2022-06-03

1 Librairies

```
library(plyr)
library(tidyverse)
library(lubridate)
library(scales)
library(ggthemes)
library(gganimate)
library(showtext)
library(plotly)
library(htmlwidgets)
```

2 Import et nettoyage des données

```
these <- as_tibble(read_csv("jeux_de_donnes/PhD_v3.csv"))
these <- subset(these, select = -c(...1))
these <- rename(these, "Vieille_discipline" = `Discipline`)
these <- rename(these, "Discipline" = `Discipline_prÃ©di`)
these <- rename(these, "Etablissement rec" = `etablissement_rec`)
these <- rename(these, "Langue rec" = `Langue_rec`)
these <- rename(these, "Années" = "Year")
these$`Date de premiere inscription en doctorat` <- dmy(these$`Date de premiere inscription en doctorat`)
these$`Date de soutenance` <- dmy(these$`Date de soutenance`)
these$`Publication dans theses.fr` <- dmy(these$`Publication dans theses.fr`)
these$`Mise a jour dans theses.fr` <- dmy(these$`Mise a jour dans theses.fr`)
these$Statut <- as.factor(these$Statut)
these$`Langue de la these` <- as.factor(these$`Langue de la these`)
these$`Accessible en ligne` <- as.factor(these$`Accessible en ligne`)
these$`Identifiant directeur` <- na_if(these$`Identifiant directeur`,"na")
these$Genre <- as.factor(these$Genre)
these$`Discipline` <- as.factor(these$`Discipline`)
levels(these$`Discipline`)[levels(these$`Discipline`) == "MathÃ©matiques"] <- "Mathématiques"
levels(these$`Discipline`)[levels(these$`Discipline`) == "Science de l'ingÃ©nieur" ] <- "Science de l'ingénieur"
```

3 Dimension esthétique de la production d'un graphe

3.1 Préparation des données

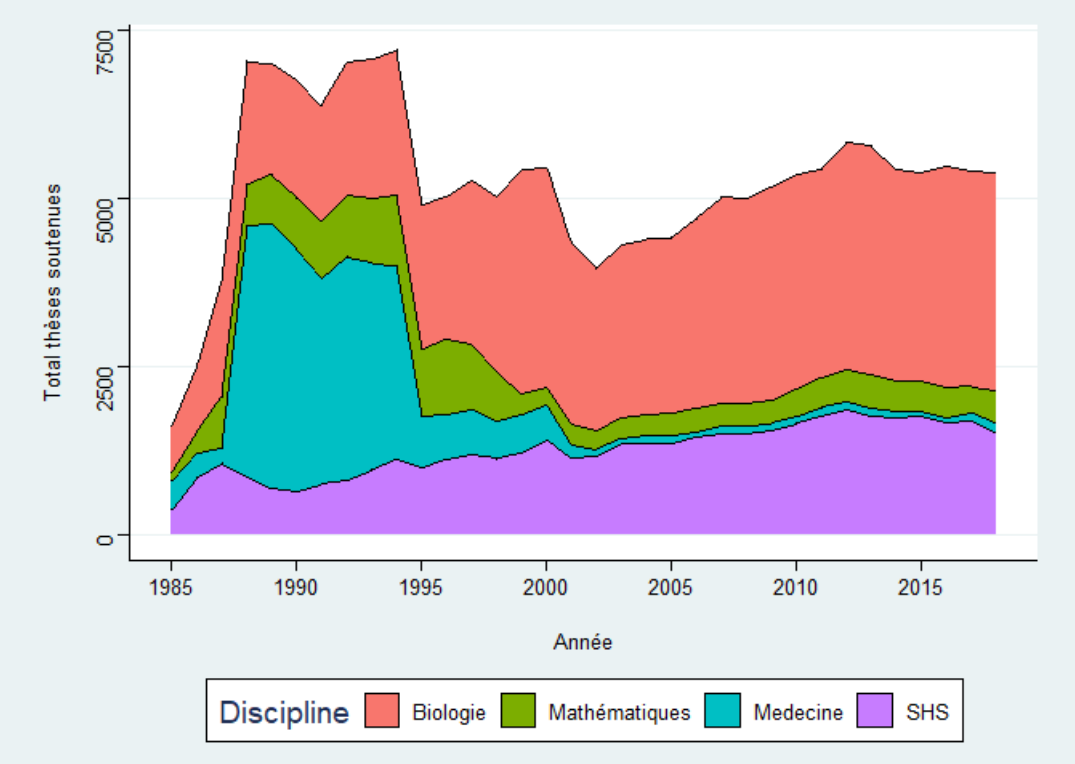
```r
# Selection des disciplines cibles pour permettre de pas surcharger les graphs
filter <- c("Biologie", "Mathématiques", "Medecine", "SHS")

# Comptage des années
these_count_year <- these %>%
  filter(Années >= 1985 & Années <= 2018 & `Discipline` %in% filter) %>%
  count(Années) %>%
  rename(count_year = n)

# Comptage des disciplines par année
these_count_year_discipline <- these %>%
  filter(Années >= 1985 & Années <= 2018 & `Discipline` %in% filter) %>%
  count(Années, `Discipline`) %>%
  rename(count_year_discipline = n)

# Jointure des tables
these_count_discipline_join <- full_join(these_count_year, these_count_year_discipline, by = "Années")

# Calcul pourcentage
these_count_discipline_join_filtered <- these_count_discipline_join %>%
  mutate(perc = count_year_discipline / count_year,
         perc = round(perc, 2),
         perc = perc * 100)
```

3.2 Exercice 1

3.2.1 Stacked area chart

Un graphique à aires empilées (stacked area chart) affiche l'évolution d'une variable numérique pour plusieurs groupes d'un ensemble de données. Chaque groupe est affiché l'un au-dessus de l'autre, ce qui facilite la lecture de l'évolution du total, mais rend difficile la lecture précise de la valeur de chaque groupe.

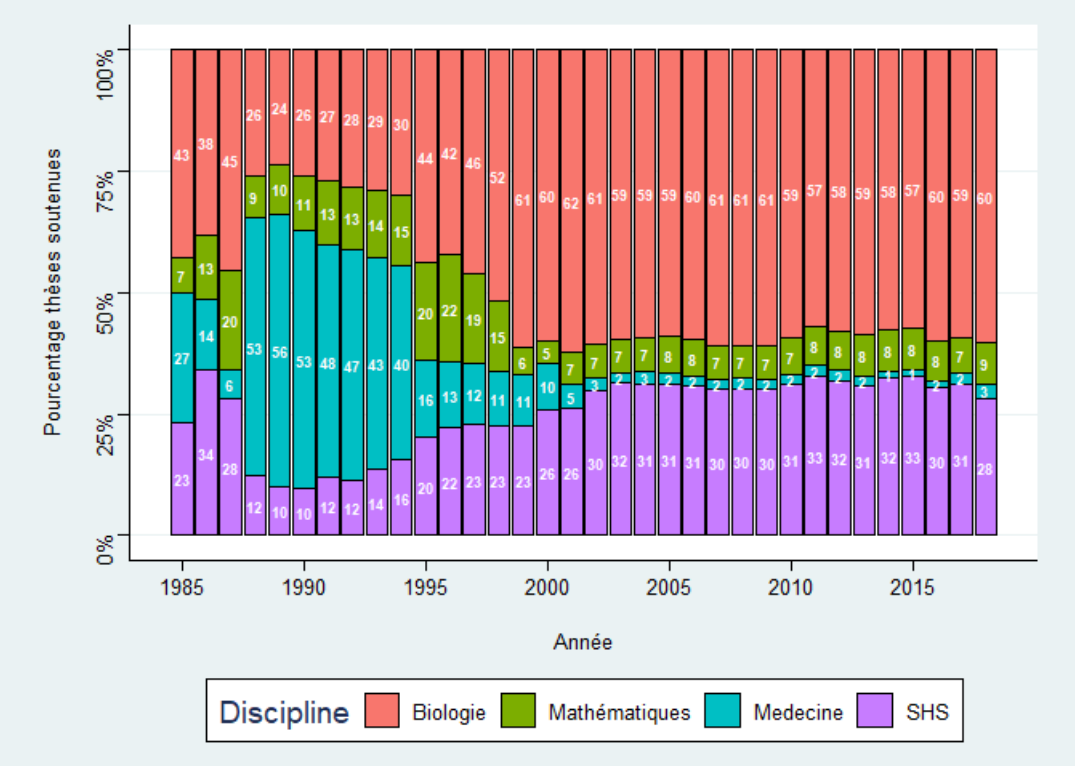
```r
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(breaks = seq(0, 20000, 2500)) +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n")
```



3.2.2 Stacked barchart

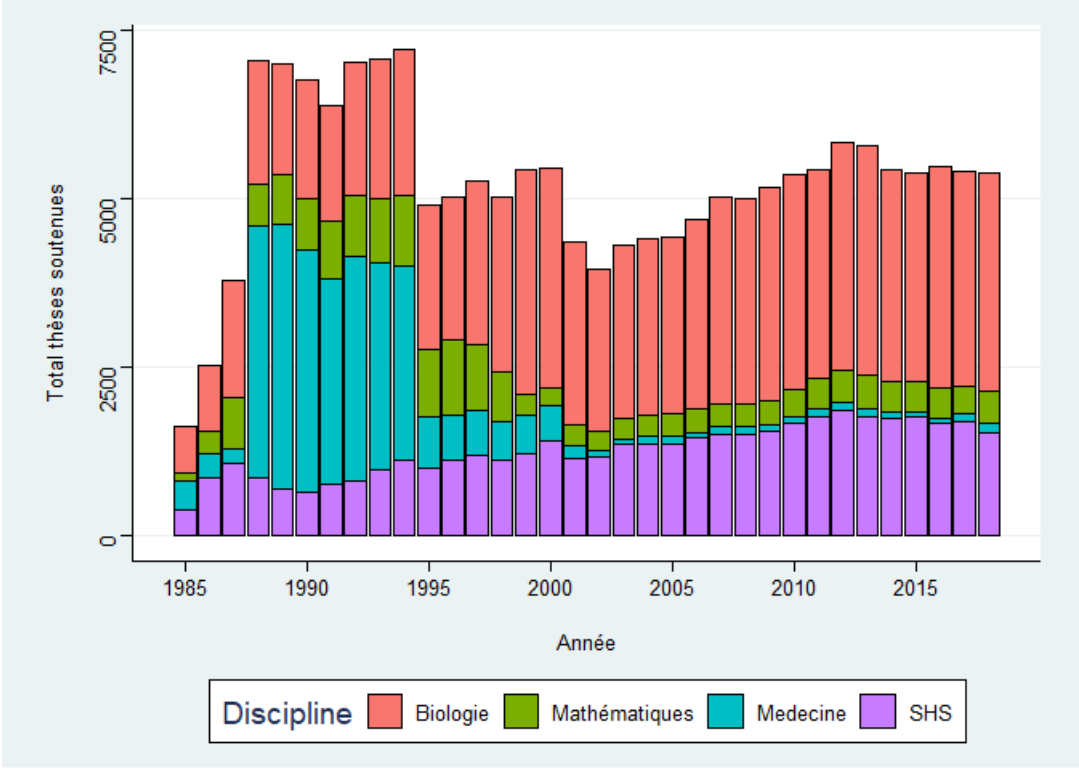
Un graphique à barres empilées (stacked bar chart) est un graphique qui utilise des barres pour montrer des comparaisons entre des catégories de données, mais avec la possibilité de décomposer et de comparer des parties d'un tout. Chaque barre du graphique représente un tout, et les segments de la barre représentent différentes parties ou catégories de ce tout.

```
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_bar(color = "black", position = "fill", stat = "identity") +
  geom_text(aes(label = perc), position = position_fill(.5),
            colour = "white", fontface = "bold", size = 2.5) +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(labels = percent_format()) +
  labs(x = "\nAnnée",
       y = "Pourcentage thèses soutenues\n")
```



3.2.3 Stacked barplot

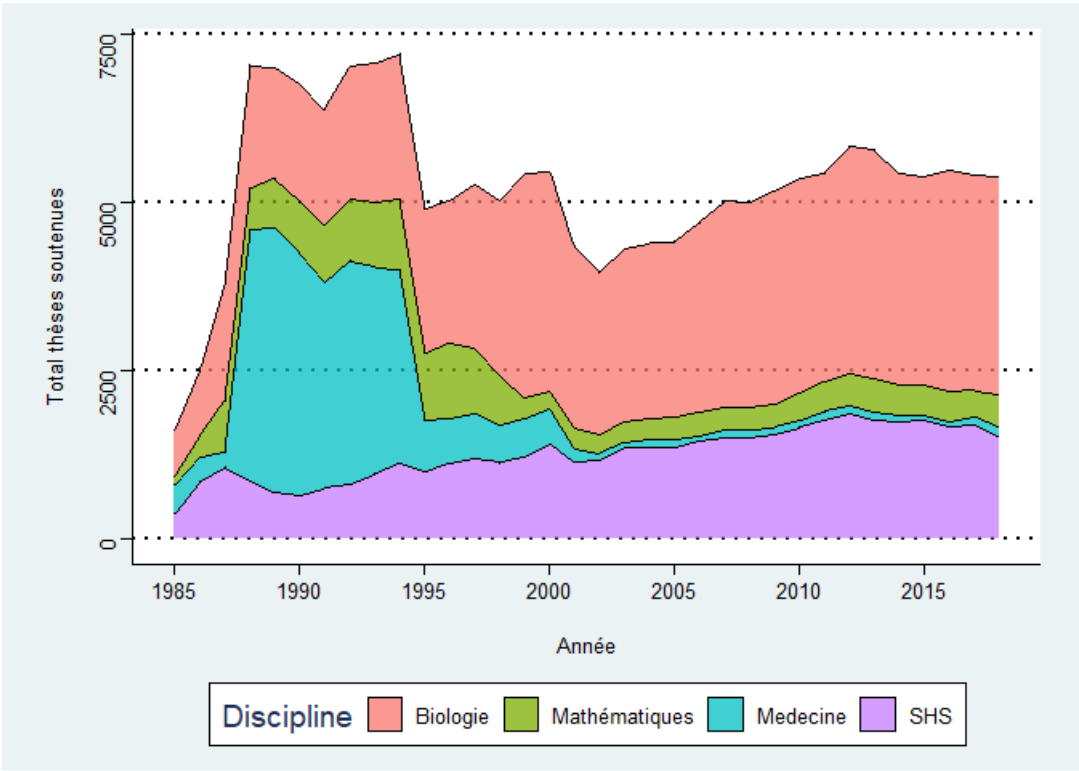
```
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_bar(color = "black", position = "stack", stat = "identity") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(breaks = seq(0, 20000, 2500)) +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n")
```



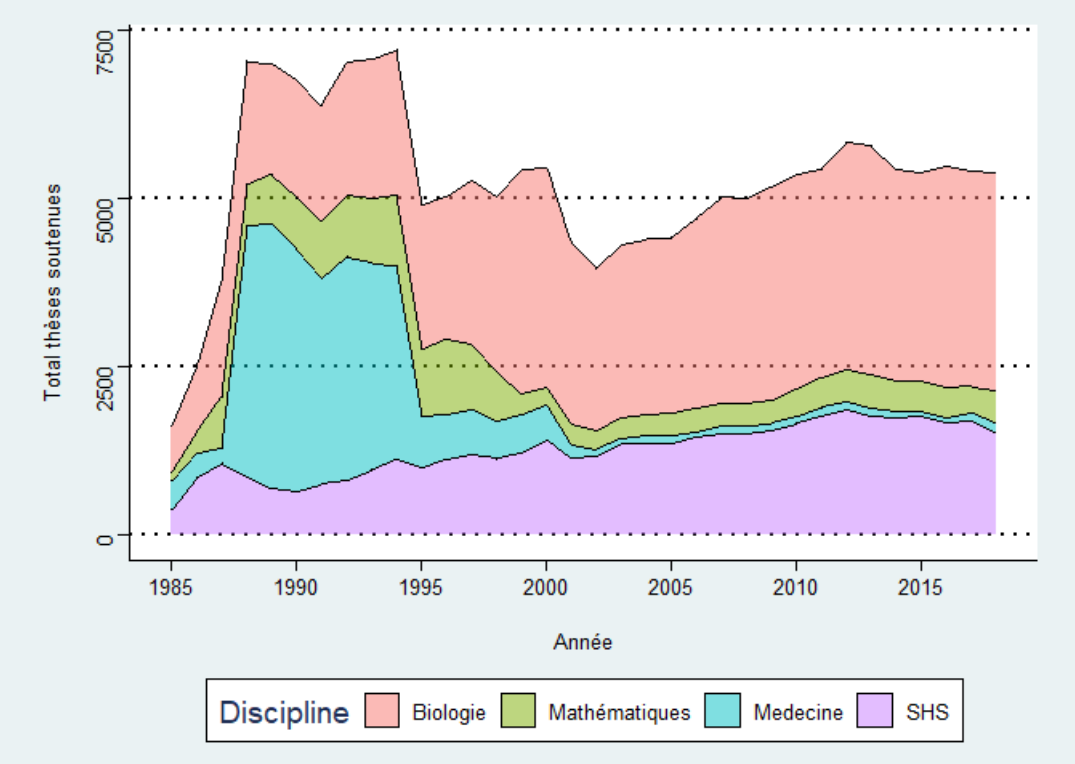
3.3 Exercice 2

La commande `alpha` permet de gérer la transparence des données du graphique.

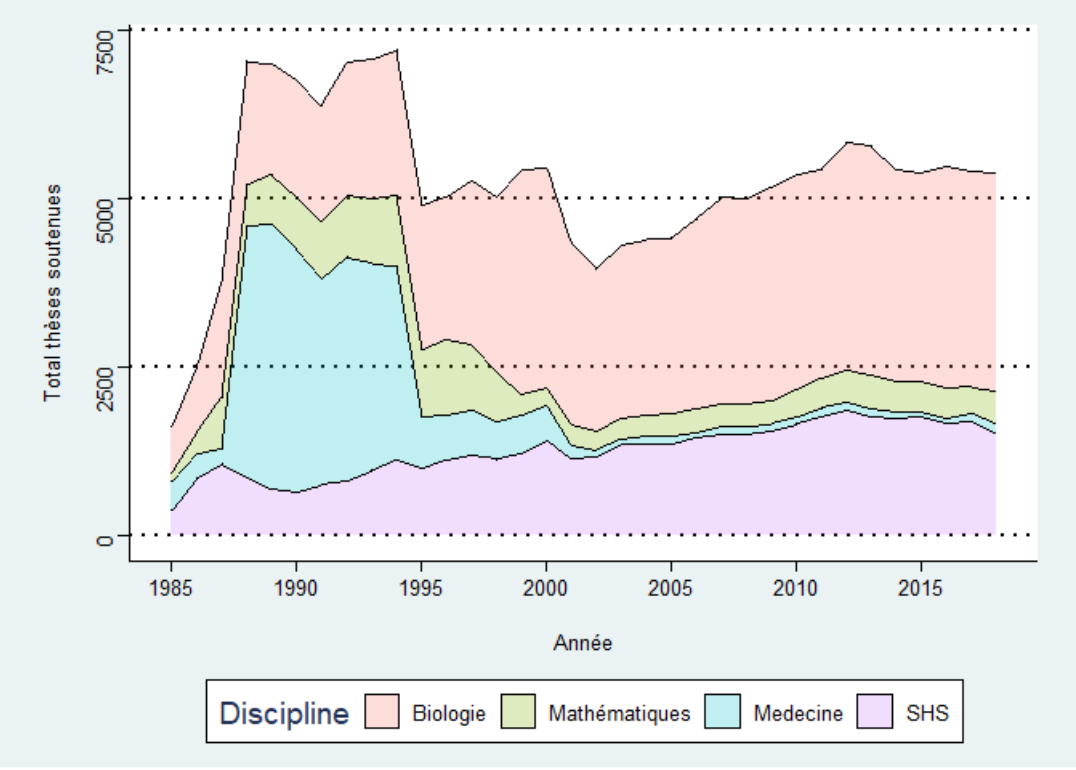
```
# alpha 0.75
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black", alpha = 0.75) +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(breaks = seq(0, 20000, 2500)) +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n") +
  theme(panel.grid.major = element_line(color = "black",
                                        size = 1,
                                        linetype = 3))
```



```r
# alpha 0.5
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black", alpha = 0.5) +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(breaks = seq(0, 20000, 2500)) +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n") +
  theme(panel.grid.major = element_line(color = "black",
                                        size = 1,
                                        linetype = 3))
```



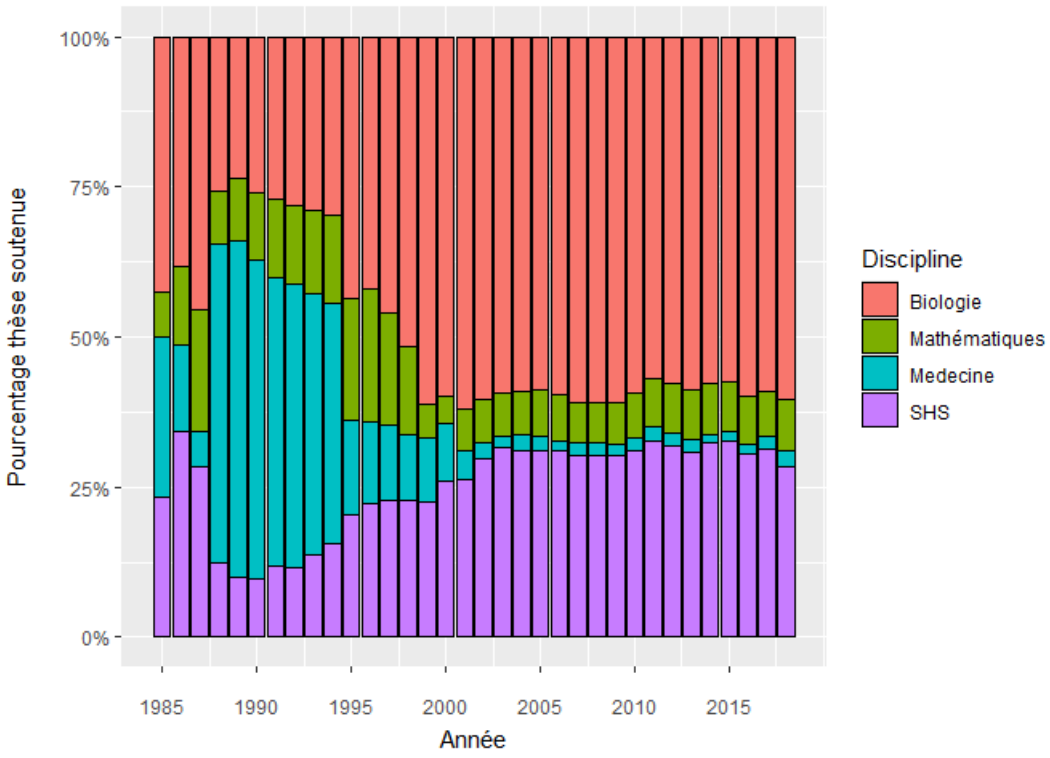

```r
# alpha 0.25
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black", alpha = 0.25) +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(breaks = seq(0, 20000, 2500)) +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n") +
  theme(panel.grid.major = element_line(color = "black",
                                        size = 1,
                                        linetype = 3))
```



3.4 Exercice 3

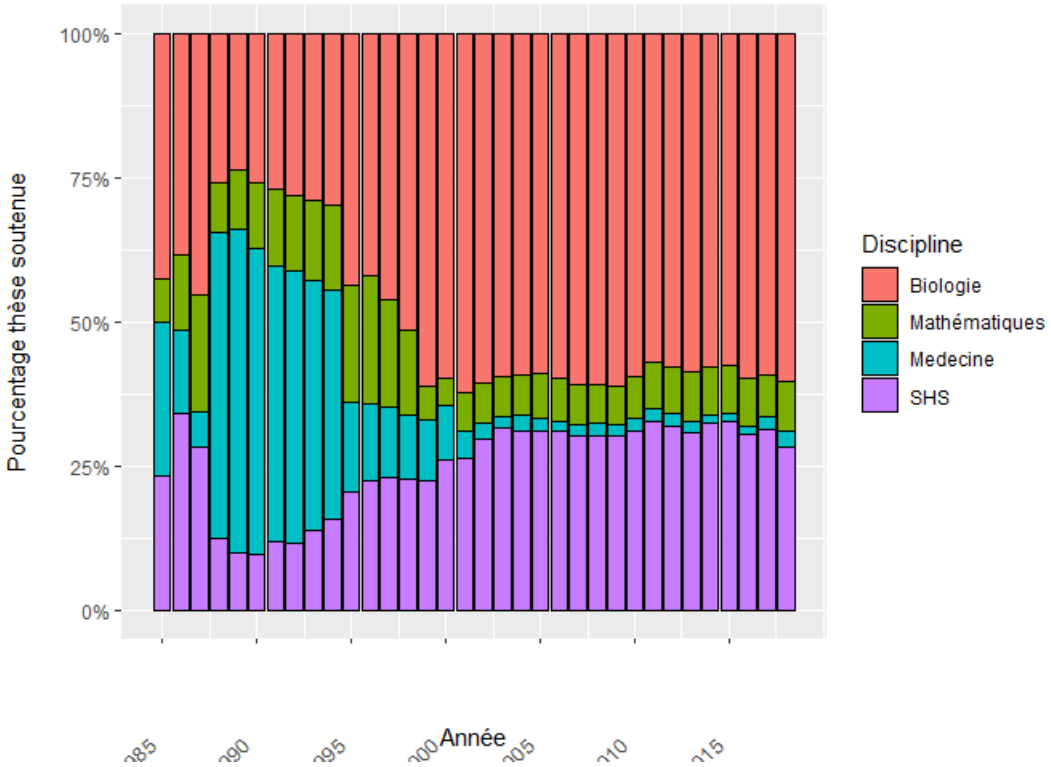
La commande `axis.text.x = element_text(vjust = -5)` permet de changer la positin vertical des labels de l'axe des abscisses.

```
# x label space
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  geom_bar(color = "black", position = "fill", stat = "identity") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(labels = percent_format()) +
  labs(x = "\nAnnée",
       y = "Pourcentage thèse soutenue\n") +
  theme(axis.text.x = element_text(vjust = -5))
```



La commande `axis.text.x = element_text(angle = 45)` permet de gérer l'angle d'affichage des labels de l'axe des abscisses.

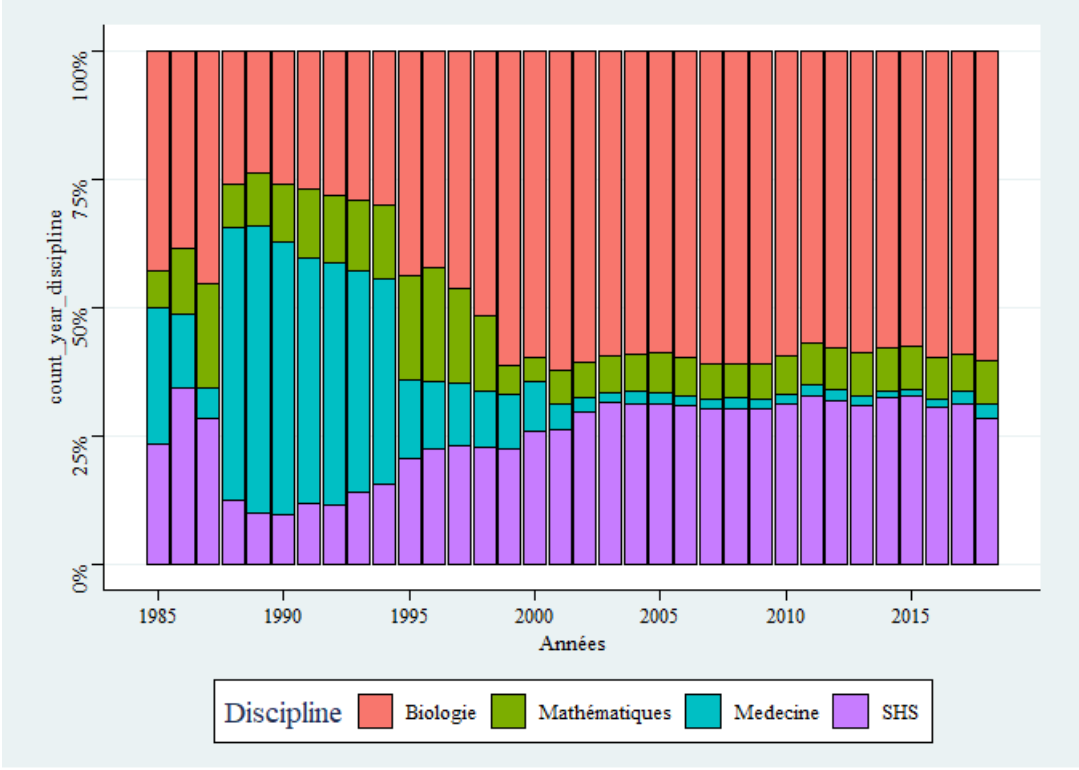
```
# x label angle
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  geom_bar(color = "black", position = "fill", stat = "identity") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(labels = percent_format()) +
  labs(x = "\nAnnée",
       y = "Pourcentage thèse soutenue\n") +
  theme(axis.text.x = element_text(angle = 45, vjust = -2))
```



3.5 Exercice 4

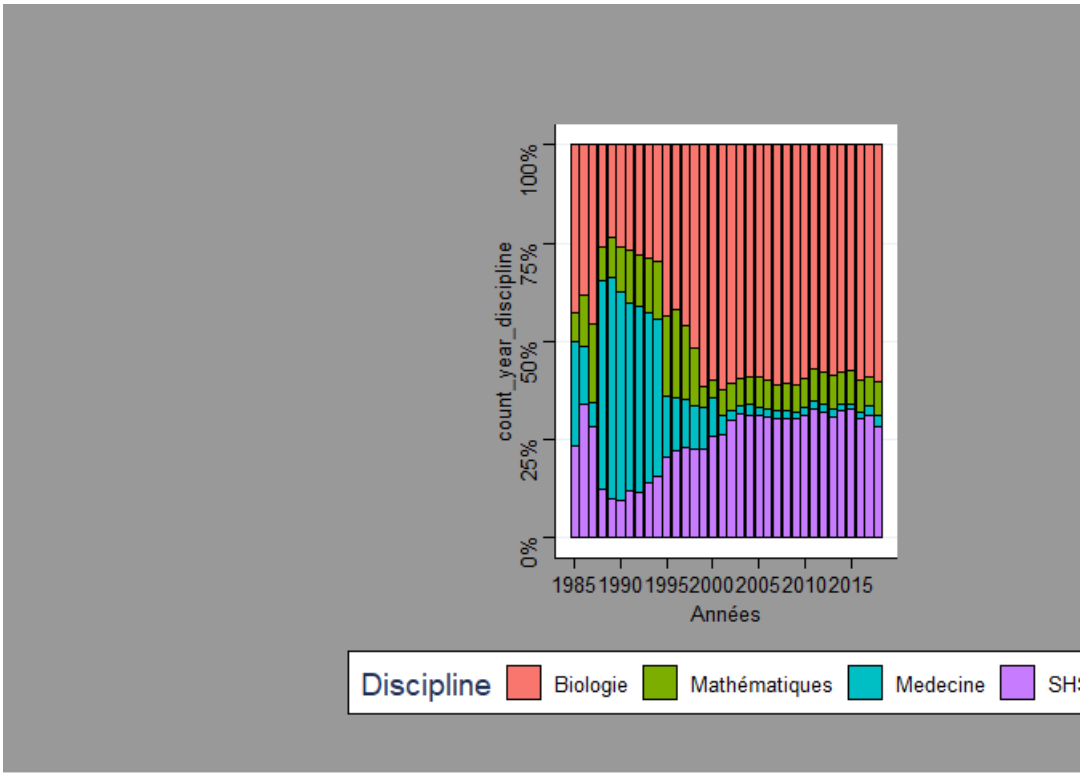
La commande `text = element_text(family = "serif")` permet de changer la police d'écriture du graphique, ici en `serif`

```
# font
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_bar(color = "black", position = "fill", stat = "identity") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(labels = percent_format()) +
  theme(text = element_text(family = "serif"))
```



La commande `plot.margin = margin(t = 2, r = 3, b = 1, l = 8, unit = "cm")` permet de gérer les tailles des marges.

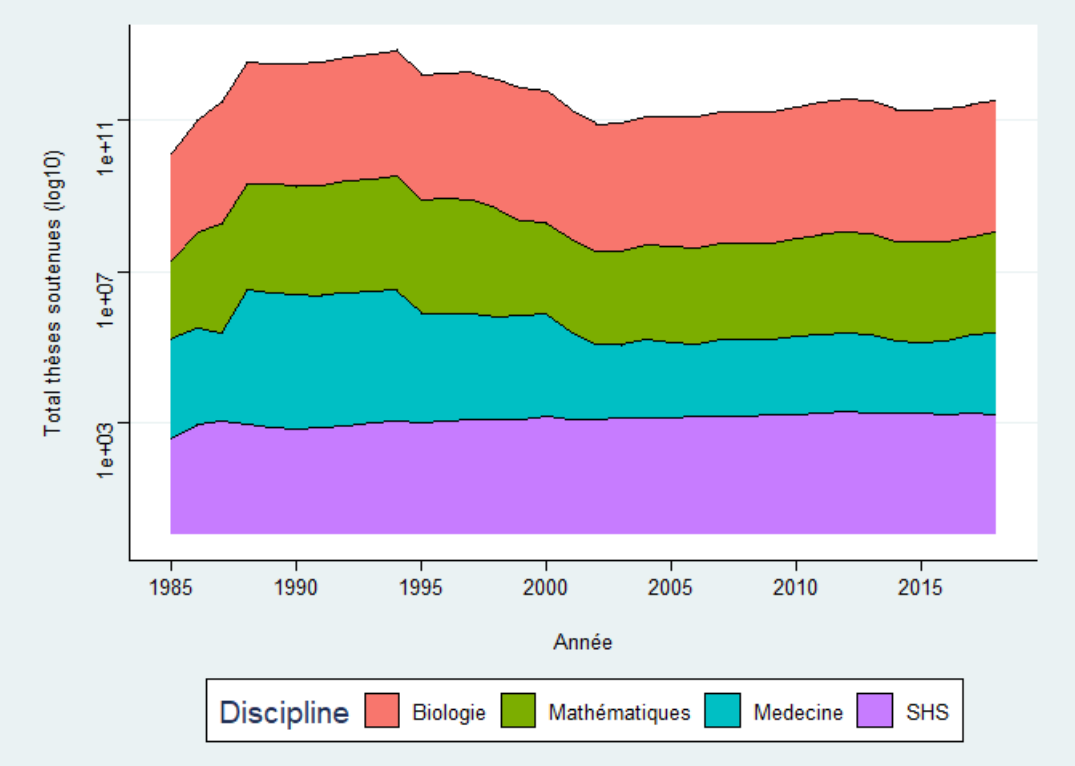
```
# margin
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_bar(color = "black", position = "fill", stat = "identity") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_continuous(labels = percent_format()) +
  theme(text = element_text(family = "Times New Roman"),
        plot.background = element_rect(fill = "gray60"),
        plot.margin = margin(t = 2, r = 3, b = 1, l = 8, unit = "cm"))
```



3.6 Exercice 5

La commande `scale_y_log10` permet de transformer l'échelle de l'axe des ordonnées en logarithme base 10.

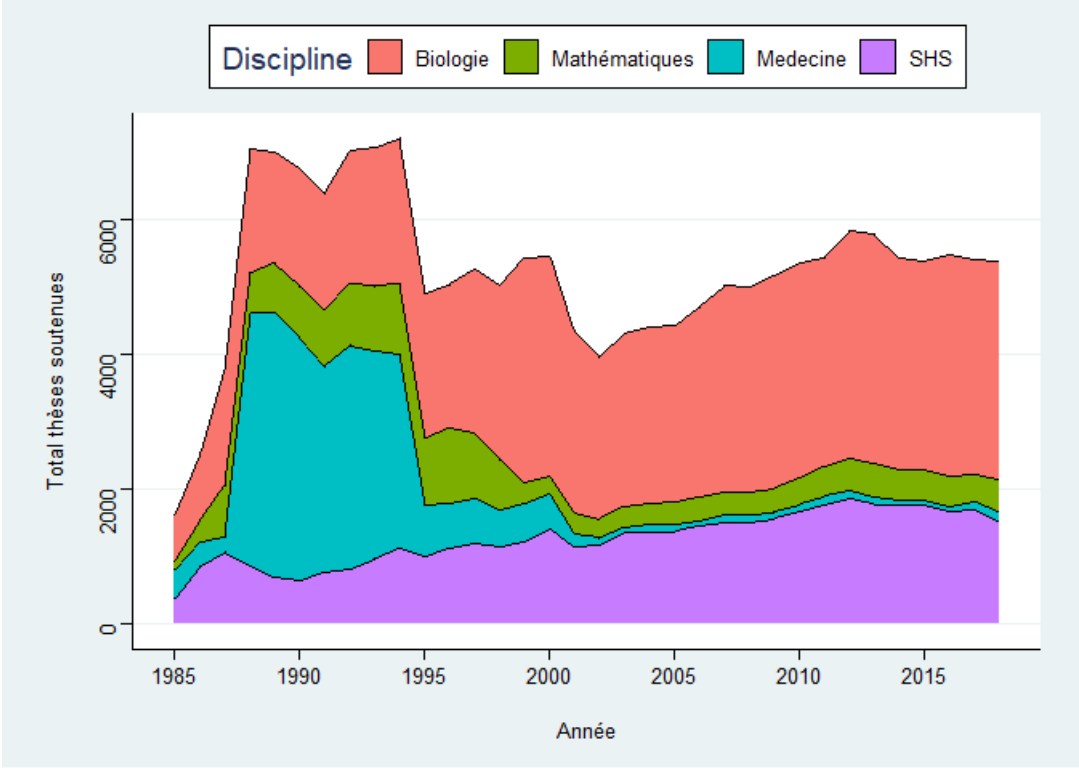
```
# y log scale
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_y_log10() +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues (log10)\n")
```



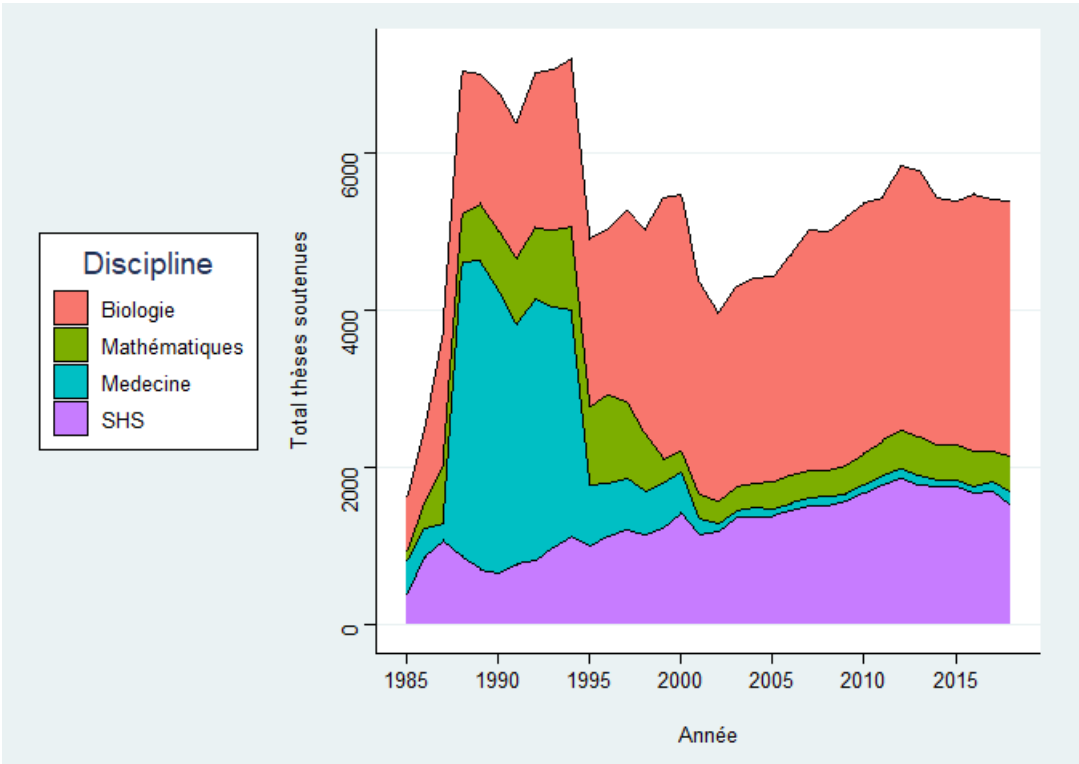
3.7 Exercice 6

Les commandes `legend.position = "top"` et `legend.position = "left"` permettent de gérer la position d'affichage de la légende, en haut et à gauche respectivement.

```
# position top
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  theme(legend.position = "top") +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n")
```



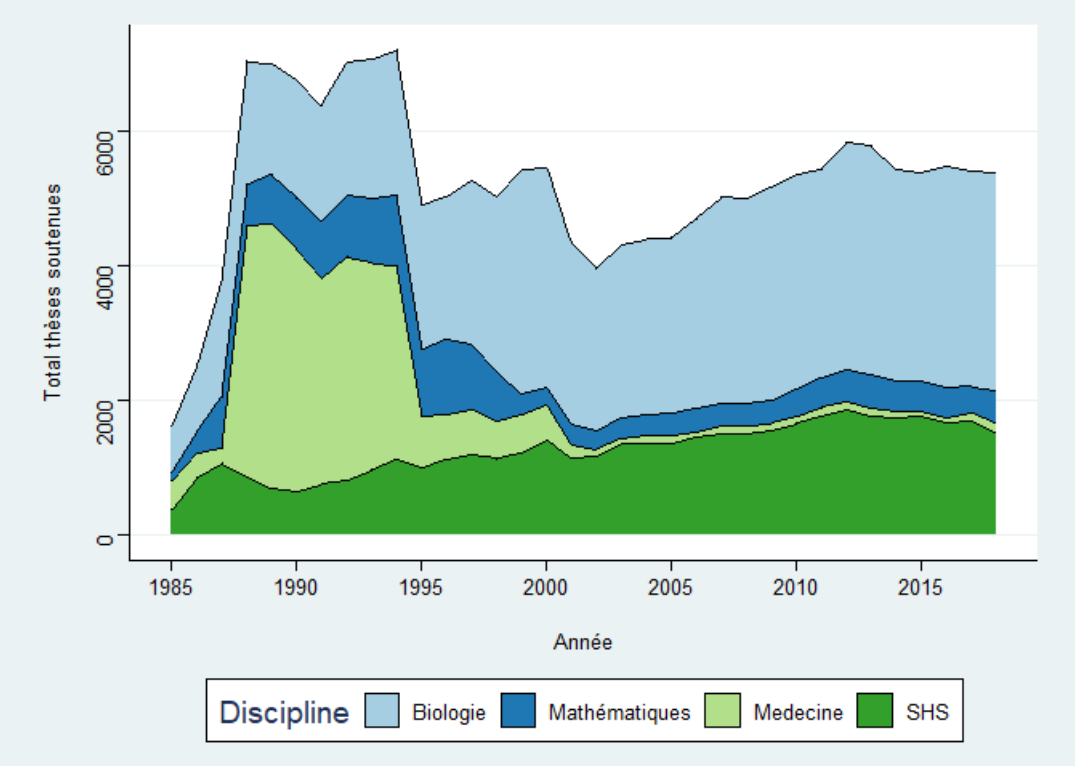

```
# position left
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  theme(legend.position = "left") +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n")
```



3.8 Exercice 7

La commande `scale_fill_brewer()` permet de changer la palette de couleur.

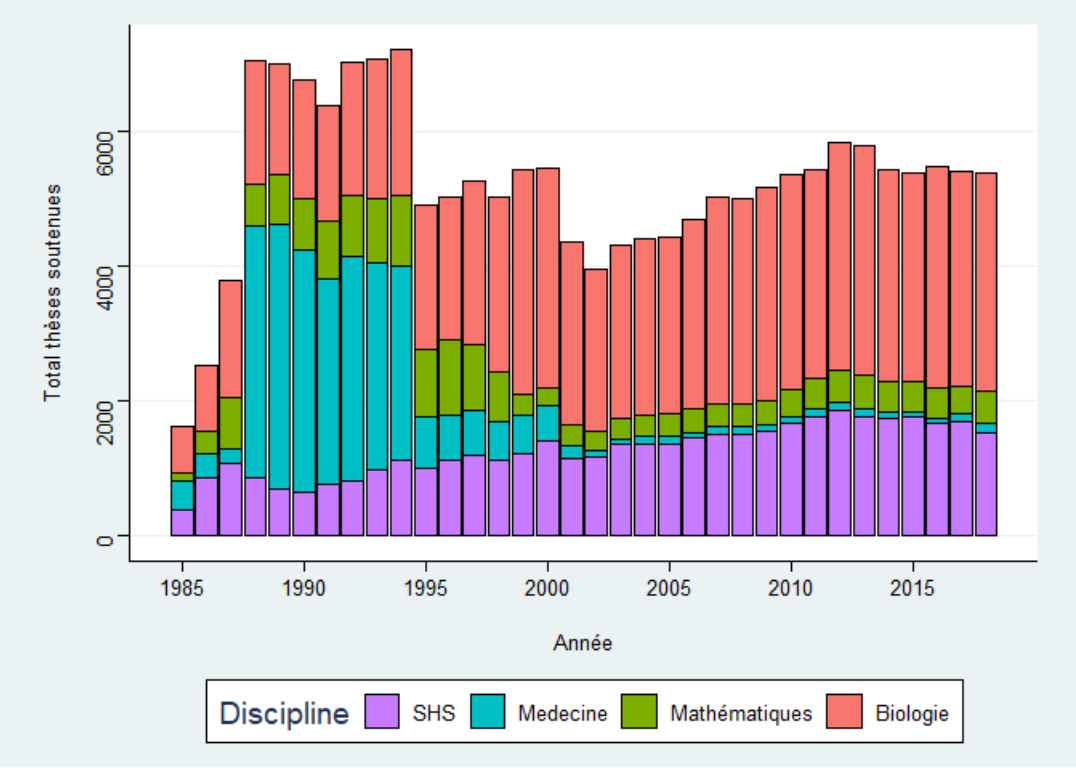
```r
# palette couleur
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_area(color = "black") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_fill_brewer(palette = "Paired") +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n")
```



3.9 Exercice 8

La commande `guide_legend(reverse = TRUE)` permet d'inverser l'ordre de la légende.

```r
# Chenger ordre légende
these_count_discipline_join_filtered %>%
  ggplot(aes(Années, count_year_discipline, fill = `Discipline`)) +
  theme_stata() +
  geom_bar(color = "black", position = "stack", stat = "identity") +
  scale_x_continuous(breaks = seq(1985, 2018, 5)) +
  scale_fill_discrete(guide = guide_legend(reverse = TRUE)) +
  labs(x = "\nAnnée",
       y = "Total thèses soutenues\n")
```



4 Production de graphes animés et interactifs

4.1 Exercice 9 : production d'un GIF

```
# Selection de la langue visée
these_langue_filtered <- these %>%
  filter(`Langue de la these` == "en" & Années >= 1985 & Années <= 2018) %>%
  select(Années, `Discipline`, `Langue de la these`) %>%
  count(Années, `Discipline`, `Langue de la these`) %>%
  rename(count = n)

these_langue_formated <- these_langue_filtered %>%
  group_by(Années) %>%
  mutate(rank = as.integer(rank(-count))) %>%
  ungroup()
```

```
static_plot <- these_langue_formated %>%
ggplot(aes(rank, group = `Discipline`, fill = `Discipline`,
           color = `Discipline`)) +
  theme_stata() +
  geom_tile(aes(y = count / 2,
            height = count,
            width = 0.9), alpha = 0.8, color = NA) +
  geom_text(aes(y = 0, label = `Discipline`), hjust = "left", colour = "black",
            fontface = "bold") +
  coord_flip(clip = "off") +
  scale_y_continuous(labels = scales::comma) +
  scale_x_reverse(breaks = seq(1, 20, 1)) +
  guides(color = FALSE, fill = FALSE) +
  theme(legend.position = "none") +
  labs(title = "Evolution des thèses soutenues en anglais par discipline au fil des ans.",
    subtitle = "Années : {closest_state}",
    x = "Rang",
    y = "Total thèses soutenues")

static_plot_animation <- static_plot +
  transition_states(Années,
                    transition_length = 2,
                    state_length = 2,
                    wrap = FALSE) +
  ease_aes('cubic-in-out')

animate(static_plot_animation, nframes = 200, fps = 25, width = 800,
        height = 600, duration = 15, end_pause = TRUE)
```

4.2 Exercice 10 : Graphique interactif contenant slider ou bouton

```
these_langue <- these
these_langue <- rename(these_langue, Langue = `Langue de la these`)
these_langue <- these_langue %>%
  mutate(Langue = as.factor(case_when(
    is.na(Langue) ~ "NA",
    Langue ==  "fr" ~ "Français",
    Langue == "en" ~ "Anglais",
    Langue == "enfr" | Langue == "fren" ~ "Bilingue",
    TRUE ~ "Autres")))
levels(these_langue$Langue)
```

```
## [1] "Anglais"  "Autres"   "Bilingue" "Français"
```

```r
# Fonction utile après
accumulate_by <- function(dat, var) {
  var <- lazyeval::f_eval(var, dat)
  lvls <- plotly:::getLevels(var)
  dats <- lapply(seq_along(lvls), function(x) {
    cbind(dat[var %in% lvls[seq(1, x)], ], frame = lvls[[x]])
  })
  dplyr::bind_rows(dats)
}


# Pourcentages des langues.
these_langue_count_year <- these_langue %>%
  select(Années) %>%
  count(Années) %>%
  rename(total_year = n)

these_langue_count_langue <- these_langue %>%
  count(Années, Langue) %>%
  rename(langue_count = n)

these_langue_fulljoin <- full_join(these_langue_count_year,
                                   these_langue_count_langue,
                                   by = "Années") %>%
  mutate(freq = round((langue_count / total_year) * 100, 3))
```

4.2.1 Graphique avec slider

```r
these_langue_accumulate <- these_langue_fulljoin %>%
  filter(Années > 2000 & Années <2019) %>%
  accumulate_by(~Années)

Noax <- list(
  title = "",
  zeroline = FALSE,
  showline = FALSE,
  showticklabels = FALSE,
  showgrid = FALSE)

tla_slider <- these_langue_accumulate %>%
  plot_ly(x = ~Années, y = ~freq,
          split = ~Langue,
          frame = ~frame,
          type = 'scatter',
          mode = 'lines',
          line = list(simplyfy = F)) %>%
  layout(xaxis = Noax)

tla_slider
```

```
htmlwidgets::saveWidget(tla_slider, file = "tla_slider.html", selfcontained = TRUE)
file.rename("tla_slider.html", "visualisation_graphique_files/html_widget/tla_slider.html")
```

```
## [1] TRUE
```

4.2.2 Graphique avec range selector

```
tlfj_selector <- plot_ly(these_langue_fulljoin, x = ~Années, y = ~freq, color = ~Langue) %>%
  add_lines() %>%
  rangeslider()

tlfj_selector
```

```
htmlwidgets::saveWidget(tlfj_selector, file = "tlfj_selector.html", selfcontained = TRUE)
file.rename("tlfj_selector.html", "visualisation_graphique_files/html_widget/tlfj_selector.html")
```

```
## [1] TRUE
```

4.2.3 Graphique avec bouton

```
these_director <- these[!grepl(",", these$`Directeur de these (nom prenom)`), ]
these_director <- these_director[!grepl("@", these_director$`Directeur de these (nom prenom)`), ]

these_director <- these_director %>%
  filter(Années > 1983 & Années < 2019) %>%
  select(`Directeur de these (nom prenom)`, `Identifiant directeur`) %>%
  group_by(`Directeur de these (nom prenom)`) %>%
  mutate(total_these_diriger = n())
```

```
td_button <- plot_ly(these_director, x = ~total_these_diriger, type = "histogram")
td_button <- td_button %>%  layout(
  title = "Drop down menus - Plot type",
  xaxis = list(domain = c(0.1, 1)),
  yaxis = list(title = "y"),
  updatemenus = list(
    list(
      y = 0.8,
      buttons = list(
        list(method = "restyle",
             args = list("type", "histogram"),
             label = "Histogram"),
        list(method = "restyle",
             args = list("type", "violin"),
             label = "Violin"),
        list(method = "restyle",
             args = list("type", "box"),
             label = "Boxplot")))
  ))

td_button
```

```
htmlwidgets::saveWidget(td_button, file = "td_button.html", selfcontained = TRUE)
file.rename("td_button.html", "visualisation_graphique_files/html_widget/td_button.html")
```

```
## [1] TRUE
```

5 Visualisation de données spatialisées

5.1 Exercice 11

```r
vol <- as_tibble(read_csv("jeux_de_donnes/df_russia_2022_final.csv"))
vol$day <- ymd(vol$day)
vol$firstseen <- ymd_hms(vol$firstseen)
vol$lastseen <- ymd_hms(vol$lastseen)

vol_21 <- vol %>%
  filter(day == "2022-02-21")

vol_28 <- vol %>%
  filter(day == "2022-02-28")

# Cities
Moscou <- c(37.61, 55.75)
Paris <- c(2.35, 48.85)
Kyiv <- c(30.52, 50.45)
Berlin <- c(13.40, 52.52)
Minsk <- c(27.56, 53.89)

# Data frame cities
city <- rbind(Moscou, Paris, Kyiv, Berlin, Minsk) %>%
  as.data.frame()
colnames(city) <- c("long", "lat")
city <- city %>%
  mutate(city_name = c("Moscou", "Paris", "Kyiv", "Berlin", "Minsk"))
```

```r
geo_21 <- list(scope = 'world',
               projection = list(type = 'azimuthal equal area'),
               visible = F,
               showcountries = T,
               countrycolor = toRGB("Black"),
               showland = TRUE,
               landcolor = toRGB("gray95"))

fig_21 <- plot_geo(locationmode = 'europe')

fig_21 <- fig_21 %>% add_markers(
  data = vol_21, x = ~longitude_1, y = ~latitude_1, text = ~callsign,
  hoverinfo = "text", alpha = 0.5)

fig_21 <- fig_21 %>% add_markers(
  data = city, x = ~long, y = ~lat, alpha = 0.75, color = I("red"),
  text = ~city_name, hoverinfo = "text")

fig_21 <- fig_21 %>% add_segments(
  data = vol_21,
  x = ~longitude_1, xend = ~longitude_2,
  y = ~latitude_1, yend = ~latitude_2,
  alpha = 0.3, size = I(3))

fig_21 <- fig_21 %>% layout(
  title = 'Vol en partance et à destination de la Russie pour le 21 février 2022',
  geo = geo_21, showlegend = FALSE, height=800)

fig_21
```

```r
htmlwidgets::saveWidget(fig_21, file = "vol_21.html", selfcontained = TRUE)
file.rename("vol_21.html", "visualisation_graphique_files/html_widget/vol_21.html")
```

```r
# vol 28 février
geo_28 <- list(scope = 'world',
               projection = list(type = 'azimuthal equal area'),
               visible = F,
               showcountries = T,
               countrycolor = toRGB("Black"),
               showland = TRUE,
               landcolor = toRGB("gray95"))

fig_28 <- plot_geo(locationmode = 'europe')

fig_28 <- fig_28 %>% add_markers(
  data = vol_28, x = ~longitude_1, y = ~latitude_1, text = ~callsign,
  hoverinfo = "text", alpha = 0.5)

fig_28 <- fig_28 %>% add_markers(
  data = city, x = ~long, y = ~lat, alpha = 0.75, color = I("red"),
  text = ~city_name, hoverinfo = "text")

fig_28 <- fig_28 %>% add_segments(
  data = vol_28,
  x = ~longitude_1, xend = ~longitude_2,
  y = ~latitude_1, yend = ~latitude_2,
  alpha = 0.3, size = I(3))

fig_28 <- fig_28 %>% layout(
  title = 'Vol en partance et à destination de la Russie pour le 28 février 2022',
  geo = geo_28, showlegend = FALSE, height=800)

fig_28
```

```r
htmlwidgets::saveWidget(fig_28, file = "vol_28.html", selfcontained = TRUE)
file.rename("vol_28.html", "visualisation_graphique_files/html_widget/vol_28.html")
```