3 Analyse de la fréquence des mots et des documents: tf-idf

Une question centrale dans l'exploration de texte et le traitement du langage naturel est de savoir comment quantifier le contenu d'un document. Pouvons-nous le faire en regardant les mots qui composent le document? Une mesure de l'importance d'un mot est sa *fréquence de terme* (tf), la fréquence à laquelle un mot apparaît dans un document, comme nous l'avons vu au chapitre 1.. Il y a cependant des mots dans un document qui se répètent plusieurs fois mais peuvent ne pas être importants; en anglais, ce sont probablement des mots comme “the”, “is”, “of”, etc. Nous pourrions choisir d’ajouter des mots tels que ceux-ci à une liste de mots vides et de les supprimer avant l’analyse, mais il est possible que certains de ces mots soient plus importants dans certains documents que d’autres. Une liste de mots vides n'est pas une approche très sophistiquée pour ajuster la fréquence des termes pour les mots couramment utilisés.

Une autre approche consiste à examiner la *fréquence de document inverse d'* un mot (idf), ce qui diminue le poids des mots couramment utilisés et augmente celui des mots peu utilisés dans une collection de documents. Ceci peut être combiné avec la fréquence de termes pour calculer le *tf-idf d'* un terme (les deux quantités multipliées ensemble), la fréquence d'un terme étant ajustée en fonction de la rareté de son utilisation.

> La statistique **tf-idf a** pour but de mesurer l’importance d’un mot pour un document dans une collection (ou un corpus) de documents, par exemple pour un roman dans une collection de romans ou pour un site Web dans une collection de sites Web.

C'est une quantité empirique ou une quantité heuristique; bien qu'il se soit avéré utile dans l'extraction de textes, les moteurs de recherche, etc., ses fondements théoriques sont considérés comme moins solides que par des experts en théorie de l'information. La fréquence de document inverse pour un terme donné est définie comme suit:

$$i\,d\,f\,(\,\text{terme}\,)\;=\;\ln\left(\frac{n_{\text{ documents}}}{n_{\text{ documents contenant terme}}}\right)$$

Nous pouvons utiliser les principes de nettoyage de données, décrits au chapitre 1 , pour aborder l'analyse tf-idf et utiliser des outils cohérents et efficaces pour quantifier l'importance des divers termes dans un document faisant partie d'une collection.

3.1 Fréquence des termes dans les romans de Jane Austen

Commençons par examiner les romans publiés de Jane Austen et examinons la fréquence du premier terme, puis tf-idf. Nous pouvons commencer simplement en utilisant des verbes dplyr tels que `group_by()` et `join()` . Quels sont les mots les plus couramment utilisés dans les romans de Jane Austen? (Calculons également le nombre total de mots de chaque roman ici, pour une utilisation ultérieure.)

```
library(dplyr)
library(janeaustenr)
library(tidytext)

book_words <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word, sort = TRUE)

total_words <- book_words %>%
  group_by(book) %>%
  summarize(total = sum(n))

book_words <- left_join(book_words, total_words)

book_words
```

```
## # A tibble: 40,379 x 4
##    book              word      n  total
##    <fct>             <chr> <int>  <int>
##  1 Mansfield Park    the    6206 160460
##  2 Mansfield Park    to     5475 160460
##  3 Mansfield Park    and    5438 160460
##  4 Emma              to     5239 160996
##  5 Emma              the    5201 160996
##  6 Emma              and    4896 160996
##  7 Mansfield Park    of     4778 160460
##  8 Pride & Prejudice the    4331 122204
##  9 Emma              of     4291 160996
## 10 Pride & Prejudice to     4162 122204
## # … with 40,369 more rows
```

Il y a une ligne dans ce `book_words` bloc de données pour chaque combinaison mot-livre; `n` est le nombre de fois que ce mot est utilisé dans ce livre et `total` le nombre total de mots dans ce livre. Les suspects habituels sont ici avec le plus élevé `n` , "le", "et", "à", et ainsi de suite. Dans la figure 3.1 , examinons la distribution de `n/total` pour chaque roman, le nombre de fois qu'un mot apparaît dans un roman divisé par le nombre total de termes (mots) de ce roman. C'est exactement ce que la fréquence de terme est.

```
library(ggplot2)

ggplot(book_words, aes(n/total, fill = book)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0009) +
  facet_wrap(~book, ncol = 2, scales = "free_y")
```

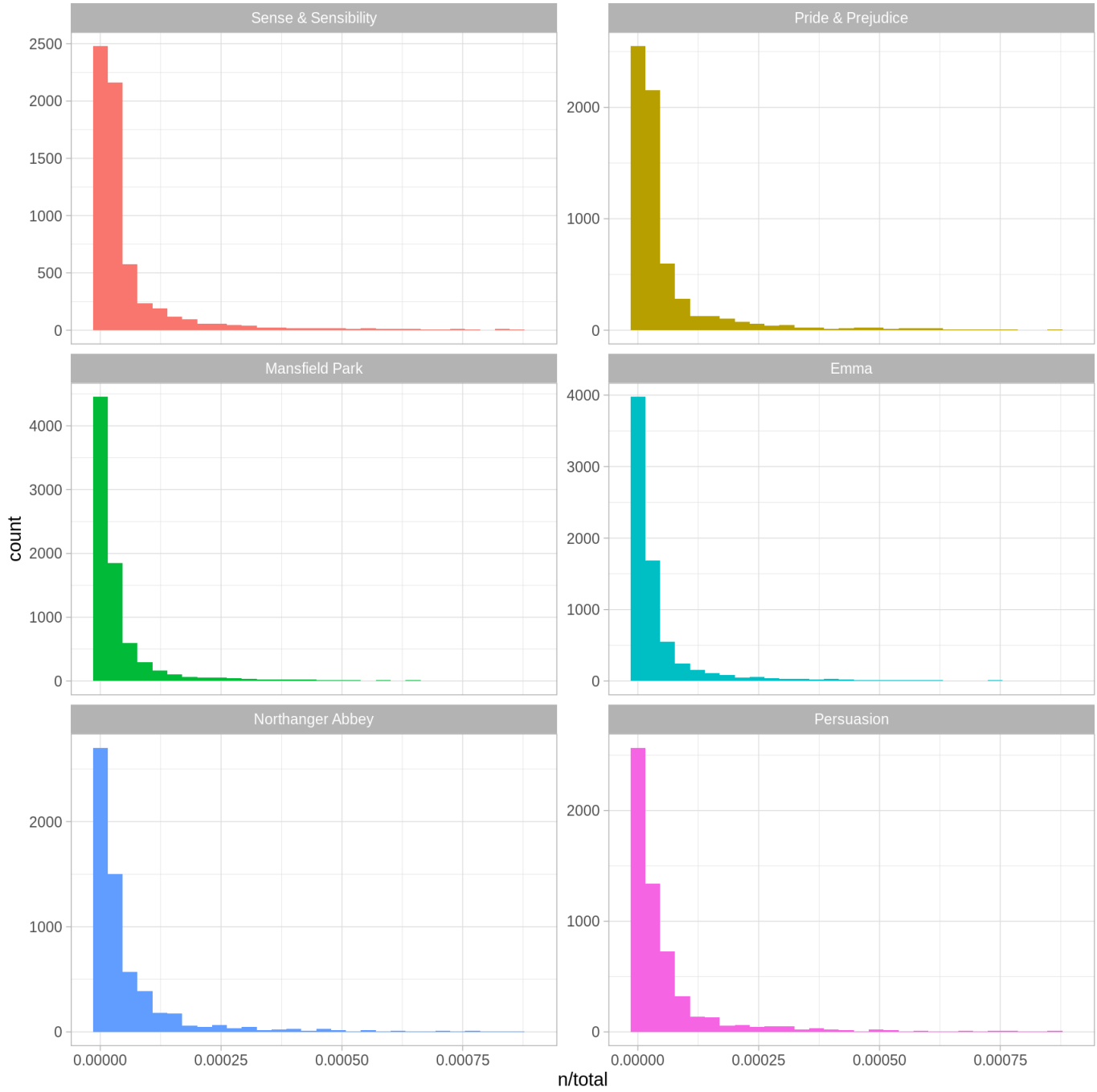

Figure 3.1: Répartition de la fréquence des termes dans les romans de Jane Austen

Il y a de très longues queues à droite pour ces romans (ces mots extrêmement rares!) Que nous n'avons pas montrés dans ces parcelles. Ces graphiques présentent des distributions similaires pour tous les romans, avec beaucoup de mots qui apparaissent rarement et moins de mots qui apparaissent fréquemment.

3.2 Loi de Zipf

Les distributions comme celles illustrées à la figure 3.1 sont typiques en langage. En fait, ces types de distributions à longue queue sont si courants dans n'importe quel corpus de langage naturel (comme un livre, beaucoup de textes d'un site Web ou des mots parlés), que la

relation entre la fréquence d'utilisation d'un mot et son rang a fait l'objet d'études; une version classique de cette relation s'appelle la loi de Zipf, d'après George Zipf, un linguiste américain du XXe siècle.

> La loi de Zipf stipule que la fréquence d'affichage d'un mot est inversement proportionnelle à son rang.

Comme nous disposons du bloc de données que nous avions utilisé pour tracer la fréquence des termes, nous pouvons examiner la loi de Zipf pour les romans de Jane Austen avec seulement quelques lignes de fonctions dplyr.

```
freq_by_rank <- book_words %>%
  group_by(book) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total)

freq_by_rank
```

```
## # A tibble: 40,379 x 6
## # Groups:   book [6]
##    book              word      n  total  rank `term frequency`
##    <fct>             <chr> <int>  <int> <int>            <dbl>
##  1 Mansfield Park    the    6206 160460     1           0.0387
##  2 Mansfield Park    to     5475 160460     2           0.0341
##  3 Mansfield Park    and    5438 160460     3           0.0339
##  4 Emma              to     5239 160996     1           0.0325
##  5 Emma              the    5201 160996     2           0.0323
##  6 Emma              and    4896 160996     3           0.0304
##  7 Mansfield Park    of     4778 160460     4           0.0298
##  8 Pride & Prejudice the    4331 122204     1           0.0354
##  9 Emma              of     4291 160996     4           0.0267
## 10 Pride & Prejudice to     4162 122204     2           0.0341
## # … with 40,369 more rows
```

La `rank` colonne ici nous indique le rang de chaque mot dans la table de fréquences; la table a déjà été commandée par `n` afin que nous puissions utiliser `row_number()` pour trouver le rang. Ensuite, nous pouvons calculer la fréquence des termes de la même manière que nous

l'avions déjà fait. La loi de Zipf est souvent visualisée en traçant le rang sur l'axe des x et la fréquence des termes sur l'axe des y, à l'échelle logarithmique. En traçant de cette façon, une relation inversement proportionnelle aura une pente négative constante.

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = book)) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```

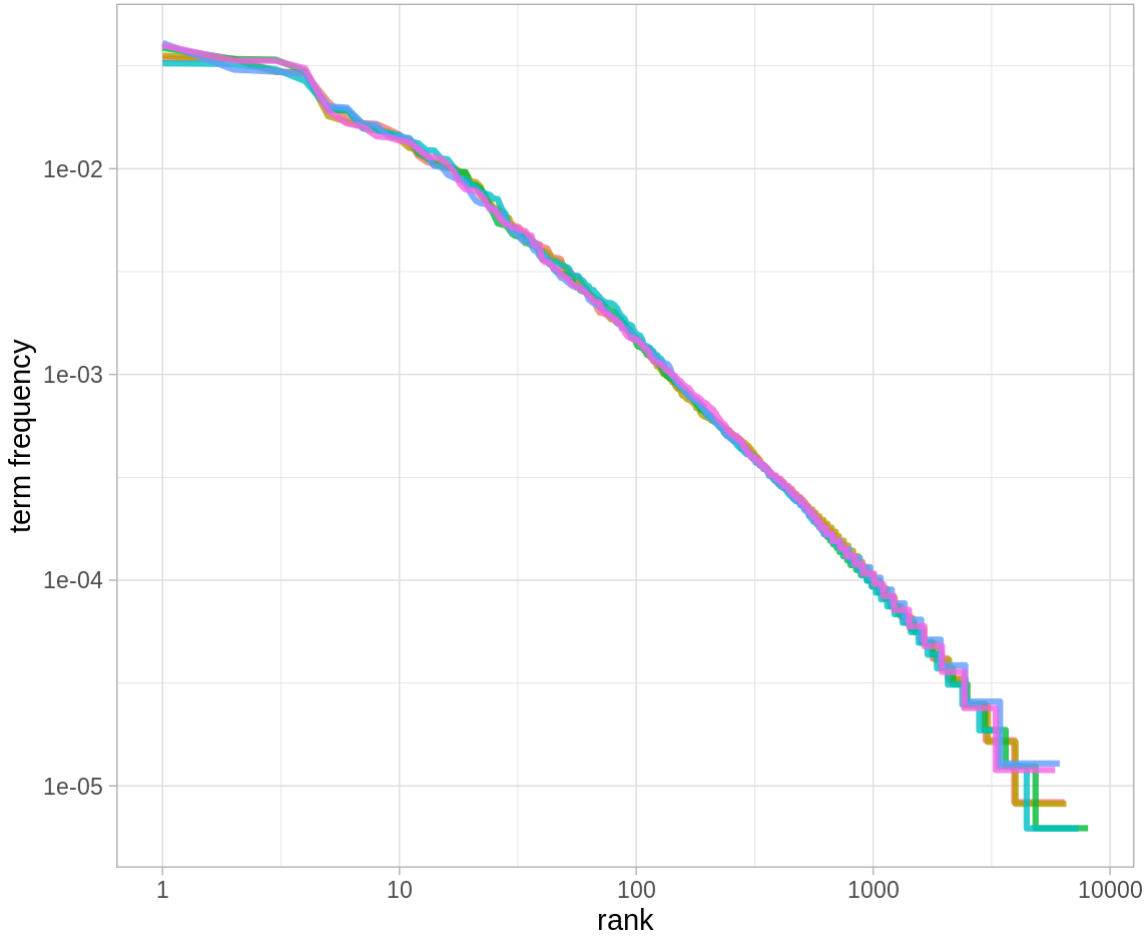

Figure 3.2: Loi de Zipf sur les romans de Jane Austen

Notez que la figure 3.2 est en coordonnées log-log. Nous constatons que les six romans de Jane Austen se ressemblent et que la relation entre rang et fréquence a une pente négative. Ce n'est pas tout à fait constant, cependant; peut-être pourrions-nous considérer cela comme une loi de force violée comportant, par exemple, trois articles. Voyons quel est l'exposant de la loi de puissance pour la partie médiane de la fourchette de rang.

```
rank_subset <- freq_by_rank %>%
  filter(rank < 500,
         rank > 10)

lm(log10(`term frequency`) ~ log10(rank), data = rank_subset)
```

```
##
## Call:
## lm(formula = log10(`term frequency`) ~ log10(rank), data = rank_subset)
##
## Coefficients:
## (Intercept)  log10(rank)
##     -0.6226      -1.1125
```

Les versions classiques de la loi de Zipf ont

$$\text{fréquence} \propto \frac{1}{\text{rang}}$$

```
freq_by_rank %>%
  ggplot(aes(rank, `term frequency`, color = book)) +
  geom_abline(intercept = -0.62, slope = -1.1, color = "gray50", linetype = 2) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```

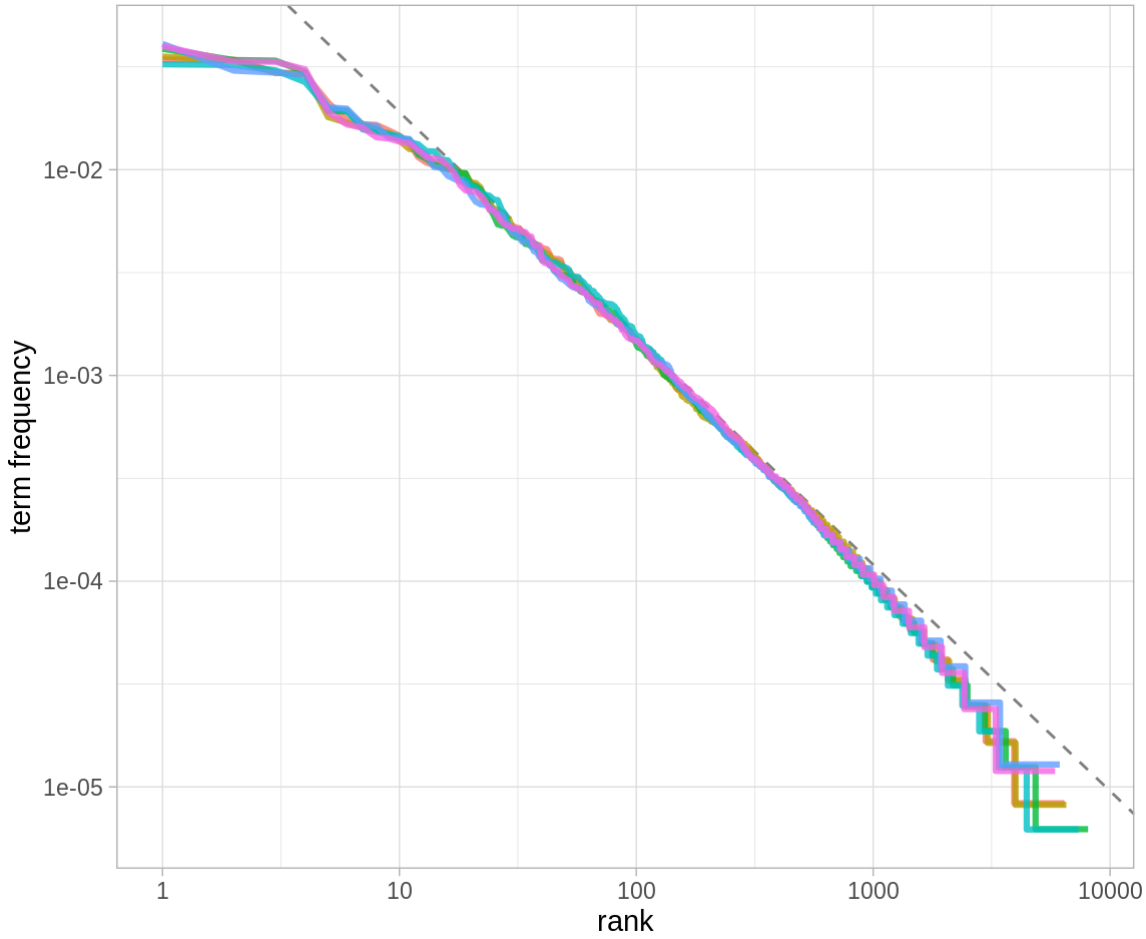

Figure 3.3: Adapter un exposant à la loi de Zipf avec les romans de Jane Austen

Nous avons trouvé un résultat proche de la version classique de la loi de Zipf pour le corpus des romans de Jane Austen. Les déviations que nous voyons ici au plus haut rang ne sont pas rares pour de nombreux types de langage; un corpus de langage contient souvent moins de mots rares que prévu par une seule loi de puissance. Les déviations au bas rang sont plus inhabituelles. Jane Austen utilise un pourcentage moins élevé des mots les plus courants que de nombreuses collections de langues. Ce type d'analyse pourrait être étendu à la comparaison des auteurs ou à toute autre collection de textes; il peut être mis en œuvre simplement en utilisant des principes de données ordonnées.

3.3 La `bind_tf_idf` fonction

L'idée de tf-idf est de trouver les mots importants pour le contenu de chaque document en diminuant le poids des mots couramment utilisés et en augmentant celui des mots peu utilisés dans une collection ou un corpus de documents, dans ce cas, l'ensemble des romans de Jane Austen. Le calcul de tf-idf tente de trouver les mots importants (c.-à-d. Communs) dans un texte, mais pas *trop* . Faisons cela maintenant.

La `bind_tf_idf` fonction du package tidytext prend en entrée un jeu de données texte ordonné avec une ligne par jeton (terme), par document. Une colonne (`word` ici) contient les termes / jetons, une colonne contient les documents (`book` dans ce cas) et la dernière colonne nécessaire contient les nombres, combien de fois chaque document contient chaque terme (`n` dans cet exemple). Nous avons calculé a `total` pour chaque livre pour nos explorations dans les sections précédentes, mais ce n'est pas nécessaire pour la `bind_tf_idf` fonction; le tableau doit seulement contenir tous les mots de chaque document.

```
book_words <- book_words %>%
  bind_tf_idf(word, book, n)

book_words
```

```
## # A tibble: 40,379 x 7
##    book              word      n  total     tf   idf tf_idf
##    <fct>             <chr> <int>  <int>  <dbl> <dbl>  <dbl>
##  1 Mansfield Park    the    6206 160460 0.0387     0      0
##  2 Mansfield Park    to     5475 160460 0.0341     0      0
##  3 Mansfield Park    and    5438 160460 0.0339     0      0
##  4 Emma              to     5239 160996 0.0325     0      0
##  5 Emma              the    5201 160996 0.0323     0      0
##  6 Emma              and    4896 160996 0.0304     0      0
##  7 Mansfield Park    of     4778 160460 0.0298     0      0
##  8 Pride & Prejudice the    4331 122204 0.0354     0      0
##  9 Emma              of     4291 160996 0.0267     0      0
## 10 Pride & Prejudice to     4162 122204 0.0341     0      0
## # … with 40,369 more rows
```

Notez que idf et donc tf-idf sont nuls pour ces mots extrêmement courants. Ce sont tous des mots qui apparaissent dans les six romans de Jane Austen. Le terme idf (qui sera alors le log naturel de 1) est égal à zéro. La fréquence de document inverse (et donc tf-idf) est très faible (proche de zéro) pour les mots figurant dans de nombreux documents d'une collection; C'est ainsi que cette approche diminue le poids des mots courants. La fréquence de document inverse sera un nombre plus élevé pour les mots qui apparaissent dans moins de documents de la collection.

Regardons les termes avec tf-idf dans les œuvres de Jane Austen.

```
book_words %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 40,379 x 6
##    book                word          n      tf   idf  tf_idf
##    <fct>               <chr>     <int>   <dbl> <dbl>   <dbl>
##  1 Sense & Sensibility elinor      623 0.00519  1.79 0.00931
##  2 Sense & Sensibility marianne    492 0.00410  1.79 0.00735
##  3 Mansfield Park      crawford    493 0.00307  1.79 0.00551
##  4 Pride & Prejudice   darcy       373 0.00305  1.79 0.00547
##  5 Persuasion          elliot      254 0.00304  1.79 0.00544
##  6 Emma                emma        786 0.00488  1.10 0.00536
##  7 Northanger Abbey    tilney      196 0.00252  1.79 0.00452
##  8 Emma                weston      389 0.00242  1.79 0.00433
##  9 Pride & Prejudice   bennet      294 0.00241  1.79 0.00431
## 10 Persuasion          wentworth   191 0.00228  1.79 0.00409
## # … with 40,369 more rows
```

Nous voyons ici tous les noms propres, des noms qui sont en fait importants dans ces romans. Aucun d'entre eux ne figure dans tous les romans et ce sont des mots importants et caractéristiques pour chaque texte du corpus des romans de Jane Austen.

$$\text{dans}(\ 6\ /\ \text{une}\ )\text{dans}(\ 6\ /\ \text{deux}\ )$$

Regardons une visualisation pour ces mots tf-idf élevés dans la figure 3.4 .

```
book_words %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(book) %>%
  top_n(15) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = book)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~book, ncol = 2, scales = "free") +
  coord_flip()
```

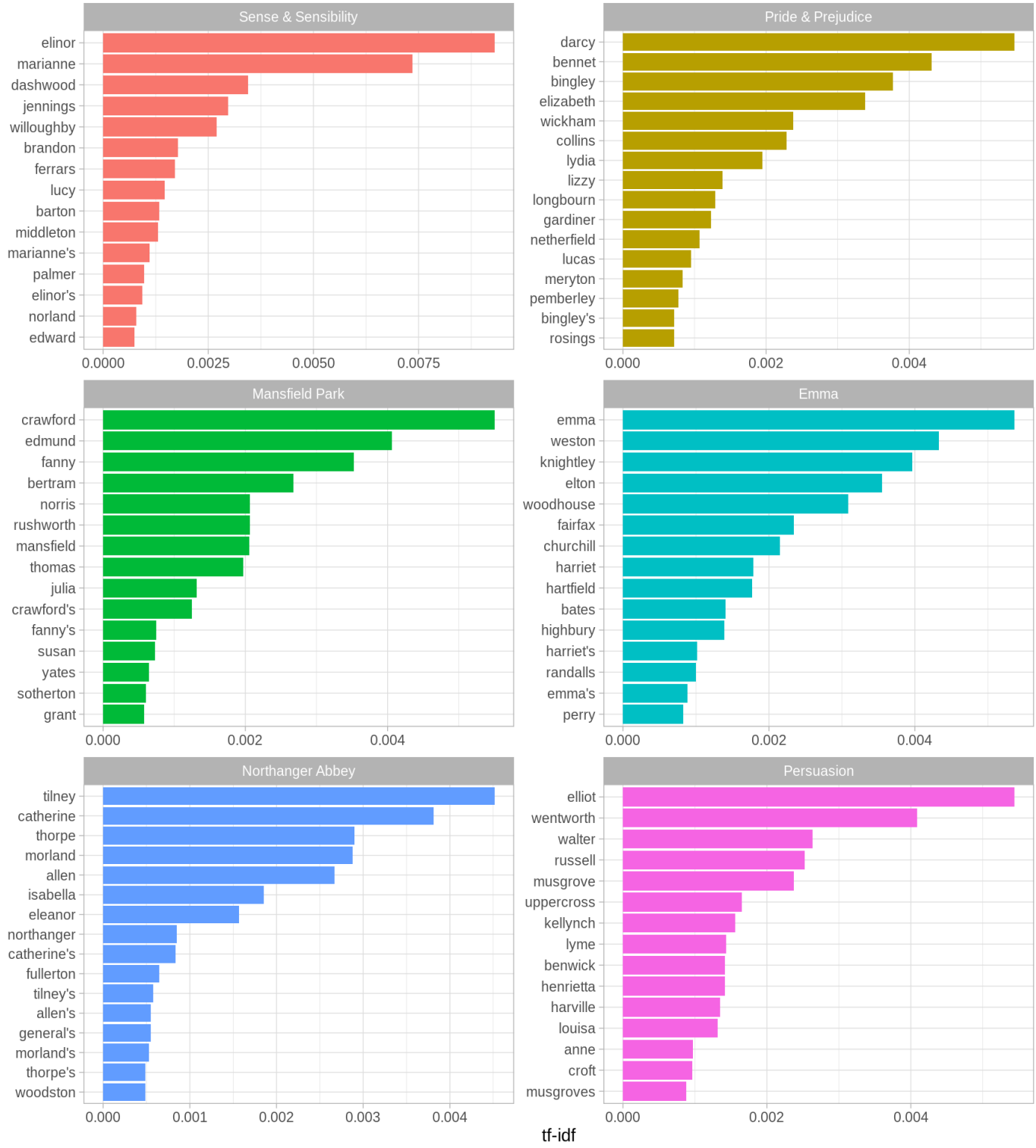

Figure 3.4: Mots tf-idf les plus élevés dans chacun des romans de Jane Austen

Toujours tous les noms propres à la figure 3.4 ! Tel que mesuré par tf-idf, ces mots sont les plus importants pour chaque roman et la plupart des lecteurs seraient probablement d'accord. Mesurant ce que tf-idf a fait ici, nous montre que Jane Austen a utilisé un langage similaire dans ses six romans, et ce qui distingue un roman du reste de la collection de ses œuvres sont les noms propres, les noms des personnes et des lieux. C'est le but de tf-idf; il identifie les mots qui sont importants pour un document dans une collection de documents.

3.4 Un corpus de textes de physique

Travaillons avec un autre corpus de documents pour voir quels termes sont importants dans un ensemble de travaux différent. En fait, sortons entièrement du monde de la fiction et du récit. Téléchargeons quelques textes de physique classiques du projet Gutenberg et voyons quels termes sont importants dans ces travaux, tels que mesurés par tf-idf. Téléchargions *Discourse on Floating Bodies* de Galileo Galilei , *Treatise on Light* de Christiaan Huygens , *expériences avec des courants alternatifs de haut potentiel et de haute fréquence* de Nikola Tesla , et *relativité: la théorie spéciale et générale* d'Albert Einstein .

C'est un groupe assez diversifié. Ce sont peut-être tous des classiques de la physique, mais ils ont été écrits sur une période de 300 ans et certains ont d'abord été écrits dans d'autres langues, puis traduits en anglais. Celles-ci ne sont pas parfaitement homogènes, mais cela n'empêche pas que cet exercice soit intéressant!

```
library(gutenbergr)
physics <- gutenberg_download(c(37729, 14725, 13476, 30155),
                              meta_fields = "author")
```

Maintenant que nous avons les textes, utilisons `unnest_tokens()` et `count()` trouvons combien de fois chaque mot a été utilisé dans chaque texte.

```
physics_words <- physics %>%
  unnest_tokens(word, text) %>%
  count(author, word, sort = TRUE)

physics_words
```

```
## # A tibble: 12,671 x 3
##    author              word      n
##    <chr>               <chr> <int>
##  1 Galilei, Galileo    the    3760
##  2 Tesla, Nikola       the    3604
##  3 Huygens, Christiaan the    3553
##  4 Einstein, Albert    the    2993
##  5 Galilei, Galileo    of     2049
##  6 Einstein, Albert    of     2028
##  7 Tesla, Nikola       of     1737
##  8 Huygens, Christiaan of     1708
##  9 Huygens, Christiaan to     1207
## 10 Tesla, Nikola       a      1176
## # … with 12,661 more rows
```

Ici, nous voyons seulement les comptes bruts; nous devons nous rappeler que ces documents ont tous des longueurs différentes. Continuons et calculons tf-idf, puis visualisons les mots élevés tf-idf de la figure 3.5 .

```
library(forcats)

plot_physics <- physics_words %>%
  bind_tf_idf(word, author, n) %>%
  mutate(word = fct_reorder(word, tf_idf)) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))

plot_physics %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  ungroup() %>%
  mutate(word = reorder(word, tf_idf)) %>%
  ggplot(aes(word, tf_idf, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~author, ncol = 2, scales = "free") +
  coord_flip()
```

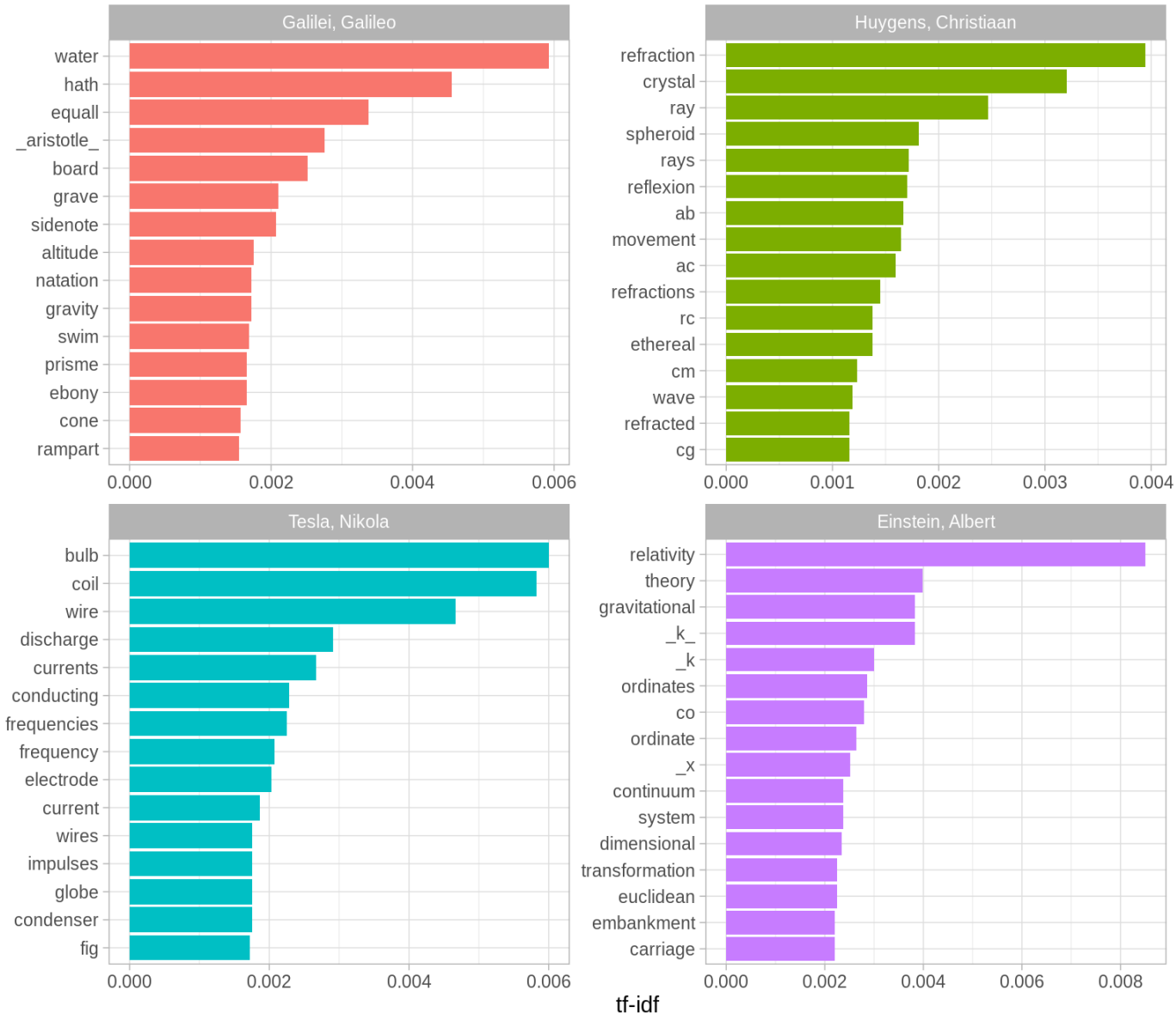

Figure 3.5: Mots tf-idf les plus élevés dans chaque texte de physique

Très intéressant en effet. Une chose que nous voyons ici est " *k* " dans le texte d'Einstein?!

```
library(stringr)

physics %>%
  filter(str_detect(text, "_k_")) %>%
  select(text)
```

```
## # A tibble: 7 x 1
##   text
##   <chr>
## 1 surface AB at the points AK_k_B. Then instead of the hemispherical
## 2 would needs be that from all the other points K_k_B there should
## 3 necessarily be equal to CD, because C_k_ is equal to CK, and C_g_ to
## 4 the crystal at K_k_, all the points of the wave CO_oc_ will have
## 5 O_o_ has reached K_k_. Which is easy to comprehend, since, of these
## 6 CO_oc_ in the crystal, when O_o_ has arrived at K_k_, because it forms
## 7 ρ is the average density of the matter and _k_ is a constant connected
```

Un peu de nettoyage du texte peut être en ordre. Notez également qu'il existe des éléments séparés «co» et «ordonnés» dans les mots tf-idf élevés pour le texte d'Einstein; la `unnest_tokens()` fonction sépare les signes de ponctuation comme des traits d'union par défaut. Notez que les scores de tf-idf pour «co» et «ordonnée» sont proches de la même chose!

“AB”, “RC”, etc. sont des noms de rayons, cercles, angles, etc. pour Huygens.

```
physics %>%
  filter(str_detect(text, "RC")) %>%
  select(text)
```

```
## # A tibble: 44 x 1
##    text
##    <chr>
##  1 line RC, parallel and equal to AB, to be a portion of a wave of light,
##  2 represents the partial wave coming from the point A, after the wave RC
##  3 be the propagation of the wave RC which fell on AB, and would be the
##  4 transparent body; seeing that the wave RC, having come to the aperture
##  5 incident rays. Let there be such a ray RC falling upon the surface
##  6 CK. Make CO perpendicular to RC, and across the angle KCO adjust OK,
##  7 the required refraction of the ray RC. The demonstration of this is,
##  8 explaining ordinary refraction. For the refraction of the ray RC is
##  9 29. Now as we have found CI the refraction of the ray RC, similarly
## 10 the ray _r_C is inclined equally with RC, the line C_d_ will
## # … with 34 more rows
```

Supprimons certains de ces mots moins significatifs pour en faire une intrigue meilleure et plus significative. Notez que nous faisons une liste personnalisée de mots vides et que nous les utilisons `anti_join()` pour les supprimer. C'est une approche flexible qui peut être utilisée dans de nombreuses situations. Nous devrons revenir en arrière quelques étapes car nous supprimons des mots du bloc de données en ordre.

```
mystopwords <- tibble(word = c("eq", "co", "rc", "ac", "ak", "bn",
                               "fig", "file", "cg", "cb", "cm",
                               "ab", "_k", "_k_", "_x"))

physics_words <- anti_join(physics_words, mystopwords,
                           by = "word")

plot_physics <- physics_words %>%
  bind_tf_idf(word, author, n) %>%
  mutate(word = str_remove_all(word, "_")) %>%
  group_by(author) %>%
  top_n(15, tf_idf) %>%
  ungroup() %>%
  mutate(word = reorder_within(word, tf_idf, author)) %>%
  mutate(author = factor(author, levels = c("Galilei, Galileo",
                                            "Huygens, Christiaan",
                                            "Tesla, Nikola",
                                            "Einstein, Albert")))

ggplot(plot_physics, aes(word, tf_idf, fill = author)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~author, ncol = 2, scales = "free") +
  coord_flip() +
  scale_x_reordered()
```

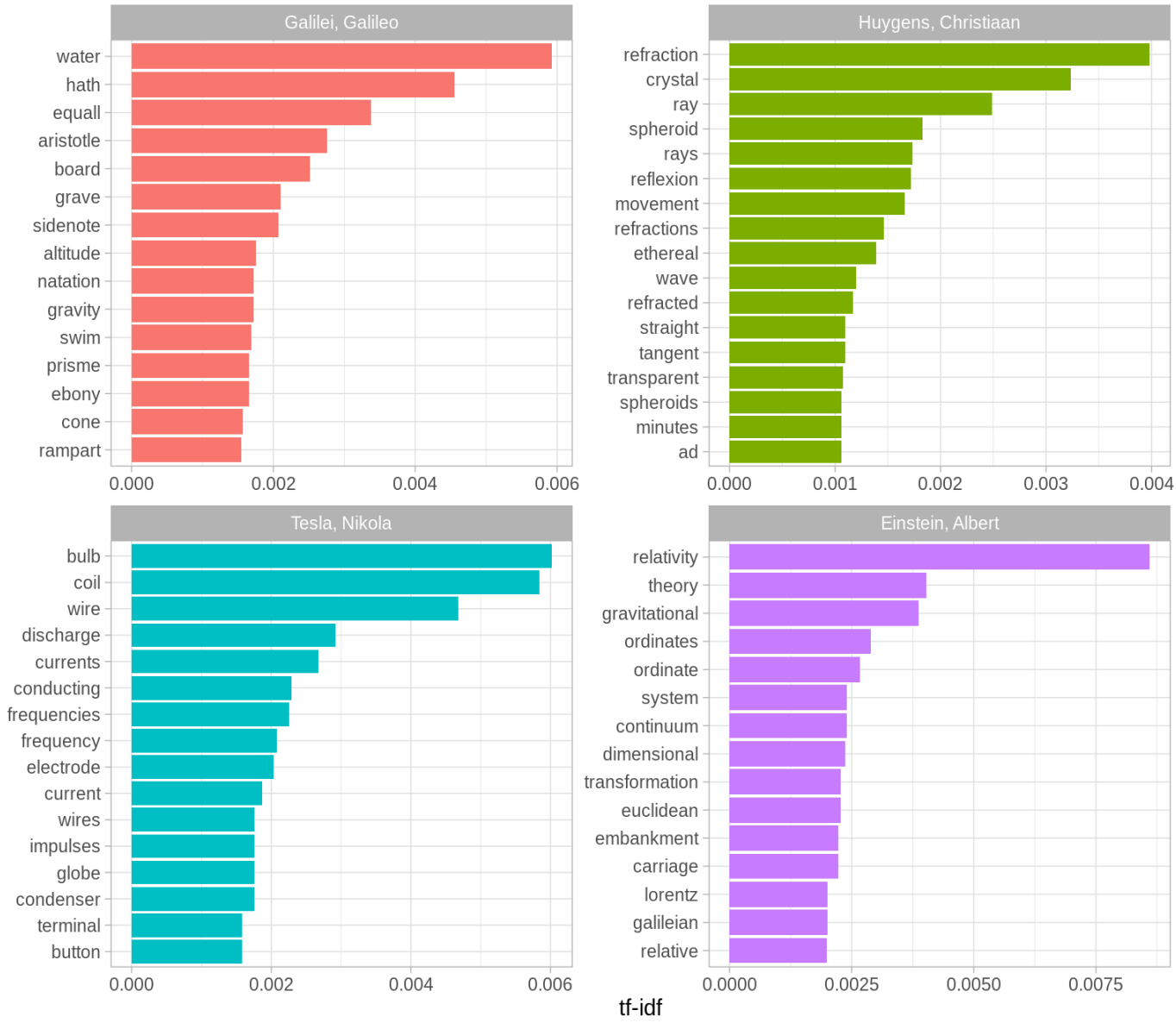

Figure 3.6: Mots tf-idf les plus élevés dans les textes de physique classiques

La figure 3.6 nous permet de conclure que nous n'entendons pas suffisamment parler de remparts ou de choses éthériques en physique aujourd'hui.

3.5 Résumé

L'utilisation de la fréquence des termes et de la fréquence inverse des documents nous permet de trouver des mots caractéristiques d'un document dans une collection de documents, qu'il s'agisse d'un nouveau texte, d'un texte physique ou d'une page Web. Explorer la fréquence des termes seule peut nous donner une idée de la façon dont le langage est utilisé dans une collection de langage naturel, et les verbes dplyr aiment `count()` et `rank()` nous donnent des outils pour raisonner sur la fréquence des termes. Le

paquet tidytext utilise une implémentation de tf-idf compatible avec les principes de données ordonnées qui nous permet de voir en quoi des mots différents sont importants dans les documents d'une collection ou d'un corpus de documents.