Customizing Plots

scales, labels, facet_wrap()

Emily Malcolm-White

```
library(tidyverse)
```

```
#Import the can_lang dataset
can_lang <- read.csv("https://raw.githubusercontent.com/ttimbers/canlang/master/inst/extda
```

A starting graph: scatterplot of `can_lang`

```
can_lang_plot <- ggplot(can_lang, aes(x=most_at_home, y=mother_tongue)) +
  geom_point() +
  xlab("Language spoken most at home \n (number of Canadian residents)") +
  ylab("Mother tongue \n (number of Canadian residents)")
```

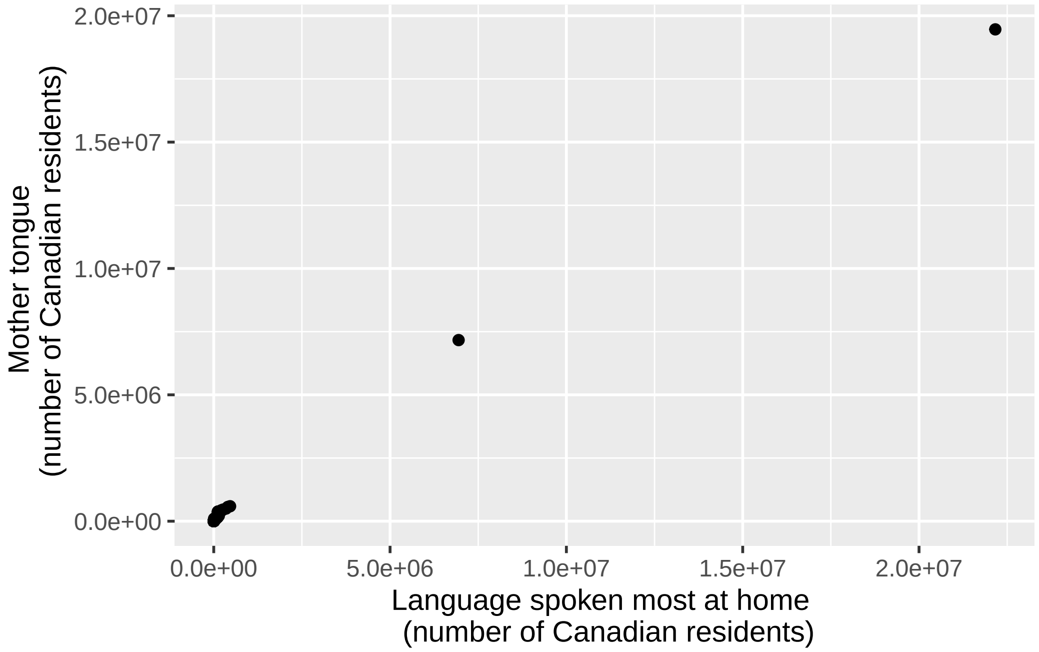
Notice anything weird about this plot?

Axis display format: `scales` package

```
 # Install the package if needed
library(scales)
```

We want to customize how the continuous x and y axes look, so we need to use the argument `labels=label_comma()` inside a `scale_*_continuous()` layer:

```
can_lang_plot
```



What other formats are available in the `scales` package?

When passing a formatting function inside `scale_*_continuous(labels = ...)` you have options!

Function	Use Case	Example Input	Example Output
`label_comma()`	Formats numbers with commas	`1234567`	`"1,234,567"`
`label_dollar()`	Formats numbers as dollar currency	`99.99`	`"$99.99"`
`label_dollar(prefix = "€")`	Formats numbers as euro currency	`99.99`	`"99.99€"`
`label_percent()`	Converts decimals to percent	`0.25`	`"25%"`
`label_pvalue()`	Formats p-values	`0.00005`	`"<0.0001"`

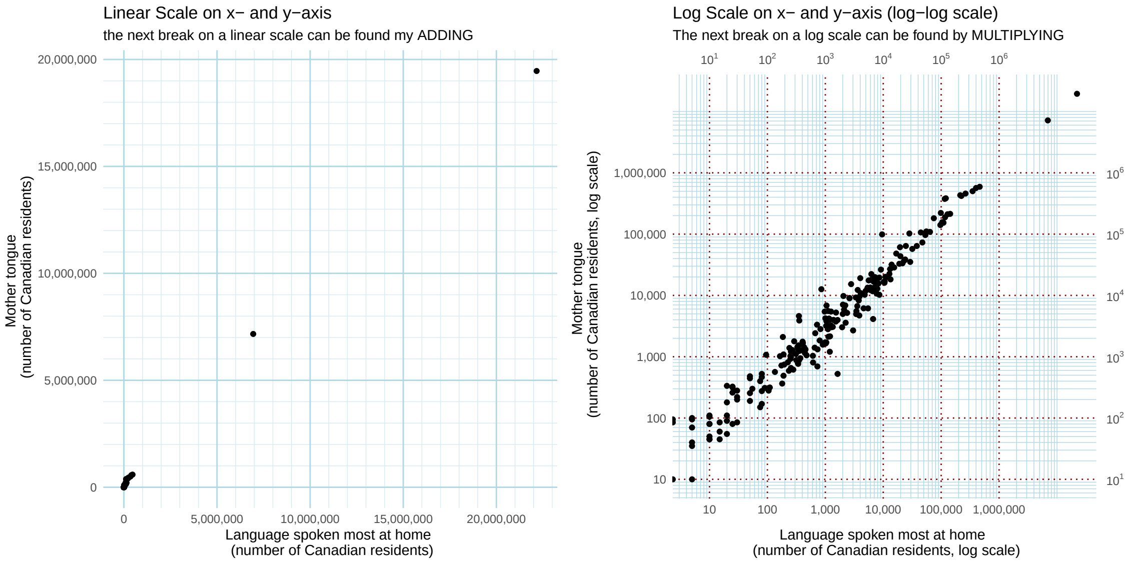
Anything else?

Logarithmic Axes Transformations

Applying a Log Transformation

When you apply a log transformation to an axis (or both axes) in a plot, you convert values using a logarithmic scale instead of a linear scale. This means:

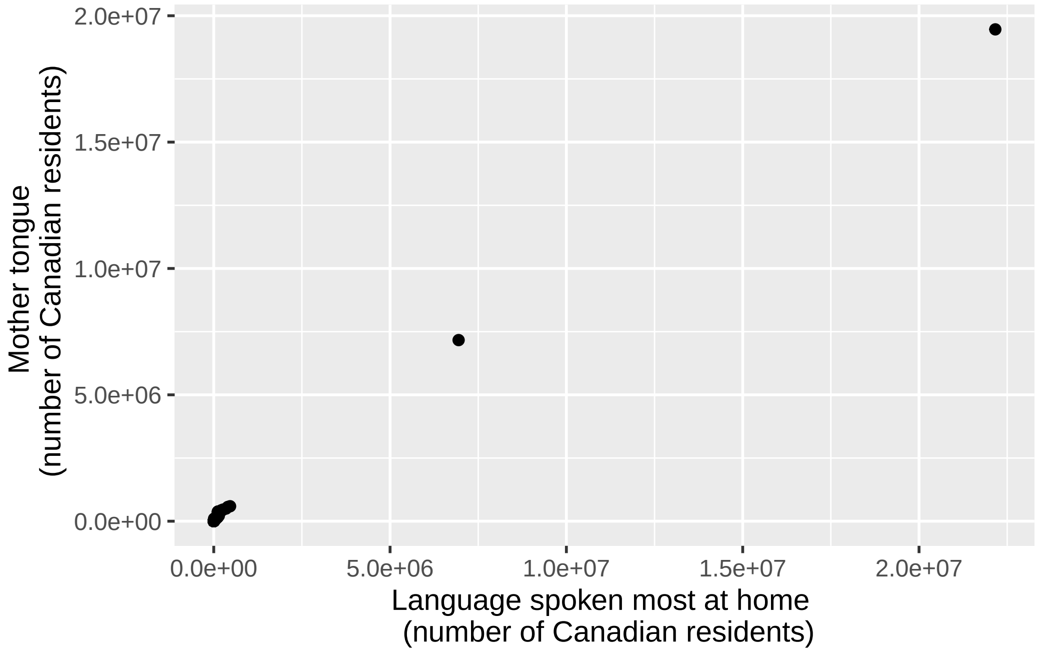
- Instead of evenly spaced values (1, 2, 3, 4, …), a logarithmic scale spaces values exponentially (1, 10, 100, 1000, …).
- The distance between ticks represents a multiplicative factor instead of an additive one.



See how much more clearly we can see all the points!

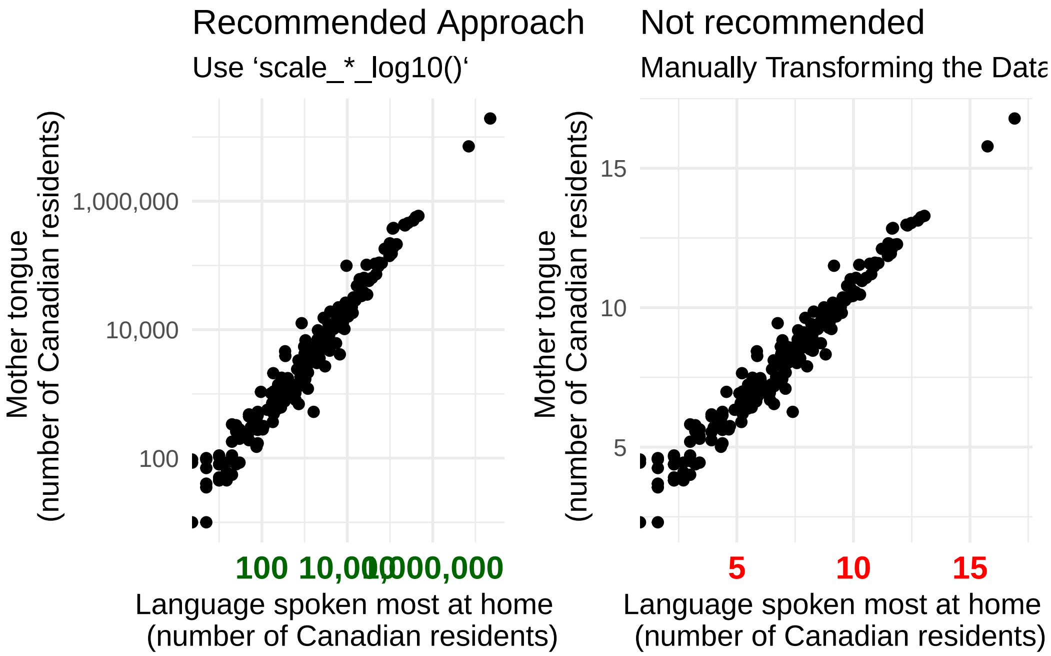
For you to do this yourself, you need to use `scale_*_log10()` instead of `scale_*_continuous()`:

```
can_lang_plot
```



1. converts x-axis to a log-scale
2. converts y-axis to a log-scale

Use `scale_*_log10()` instead of `log(variable)`



Using percents on a log scale

`mutate` to create new columns with percentage of Canadians who speak the language as their mother tongue:

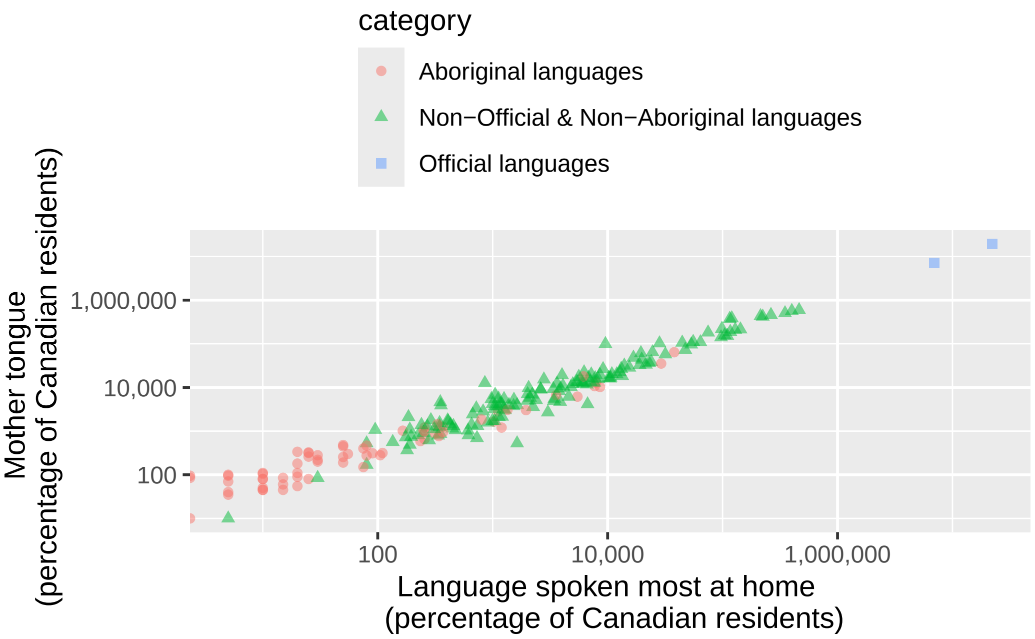
```
can_lang <- can_lang %>%
  mutate(
    mother_tongue_percent = (mother_tongue / 35151728) * 100,
    most_at_home_percent = (most_at_home / 35151728) * 100
  )
```

Scatterplot with Percents and Colors

Create a scatterplot with `most_at_home_percent` and `mother_tongue_percent`. Vary the color and shape of the points depending on the category of language. You may need to adjust the position of the legend:

```
can_lang_percent_plot <- ggplot(can_lang, aes(x = most_at_home,
                                              y = mother_tongue )) +
  geom_point(aes(color = category, shape=category), alpha=0.5) + #<3>
  xlab("Language spoken most at home \n (percentage of Canadian residents)") +
  ylab("Mother tongue \n (percentage of Canadian residents)") +
  theme(legend.position = "top", legend.direction = "vertical") + #<4>
  scale_x_log10(labels = comma) +
  scale_y_log10(labels = comma)

can_lang_percent_plot
```

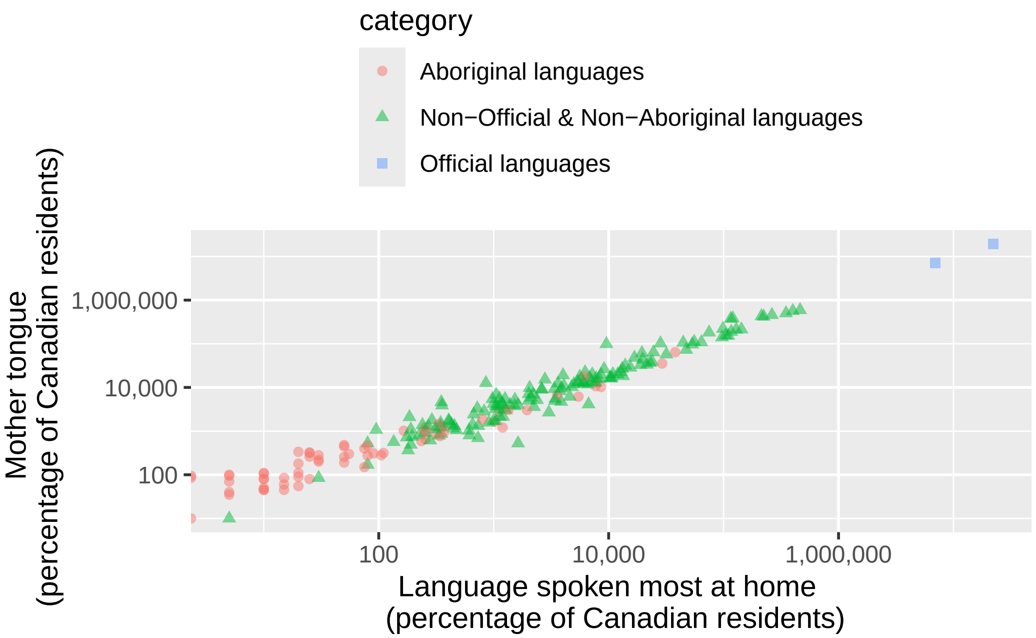


Labels

Adding text to a plot is one of the most common forms of annotation. Most plots will not benefit from adding text to every single observation on the plot, but labeling outliers and other important points is very useful.

A add label for each language in this dataset using `geom_text(aes(label = language))`:

```
can_lang_percent_plot
```



Yikes! This is way too much going on in one plot. A few options to try when this happens:

- Decrease the font size of the labels (using the `size=` argument inside `geom_text`).
- Use the `ggrepel` package to spread out the labels a bit more
- Pick out only a subset of the points to label

Using `ggrepel`

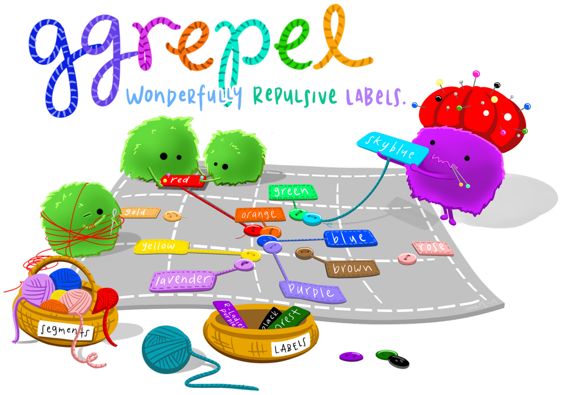
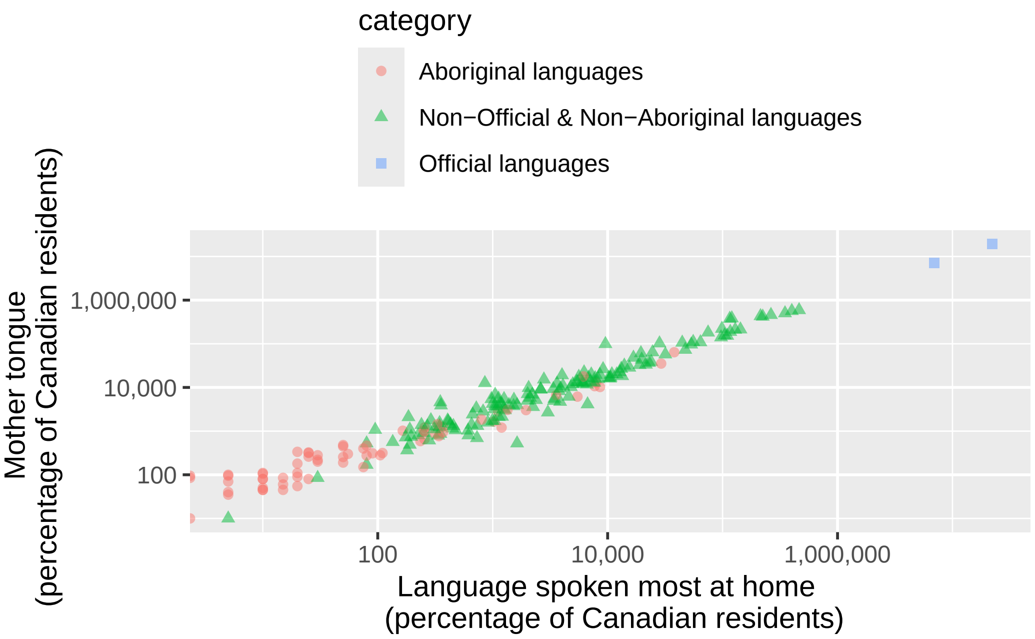


Figure 1: Artwork by @allisonhorst

```r
library(ggrepel)

can_lang_percent_plot
```

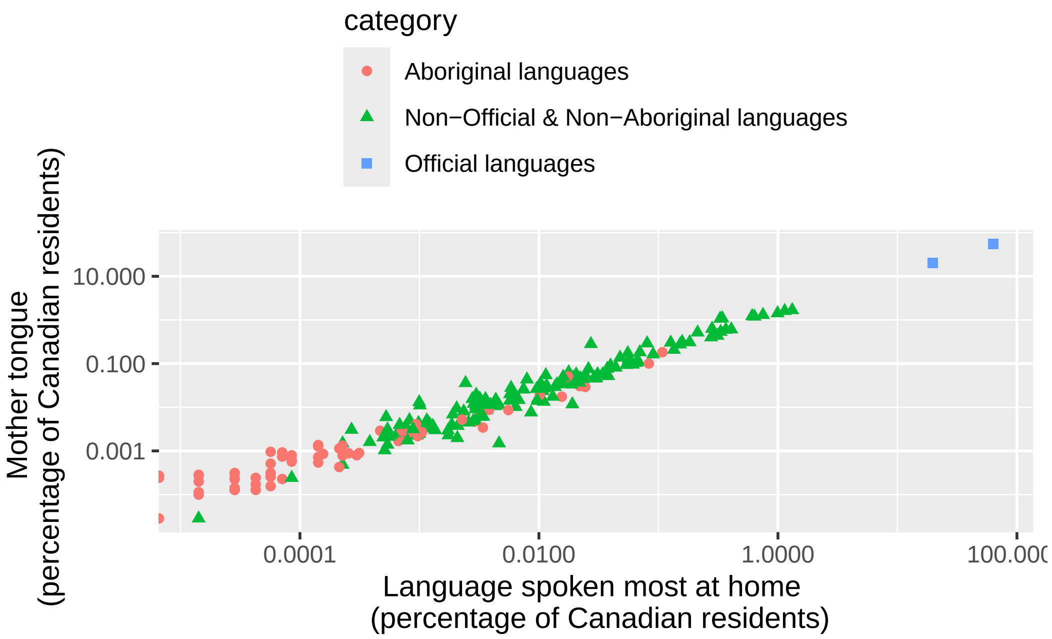


Subset the labels

Create a new column for the labels. Use `case_when` (or `ifelse`) to only use the official language names and not to put a label for other language categories.

```r
can_lang <- can_lang %>%
  mutate(official_languages = case_when(category == "Official languages" ~ language, TRUE
```

```r
can_lang_percent_plot
```



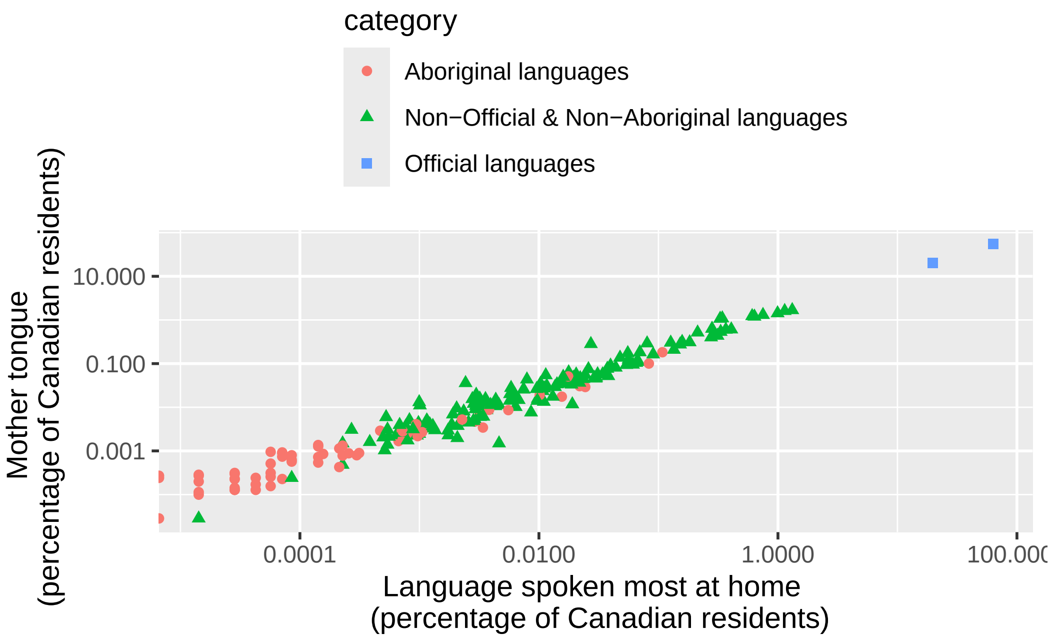
```
# We need to redo the base plot with the new can_lang dataset with the new official_langua
can_lang_percent_plot <- ggplot(can_lang, aes(x = most_at_home_percent,  y = mother_tongue
  geom_point(aes(color = category, shape=category)) +
  xlab("Language spoken most at home \n (percentage of Canadian residents)") +
  ylab("Mother tongue \n (percentage of Canadian residents)") +
  theme(legend.position = "top", legend.direction = "vertical") +
  scale_x_log10(labels = comma) +
  scale_y_log10(labels = comma)
```

Facet Wrap

`facet_wrap()` is a function in the `ggplot2` package that allows you to create a multi-panel plot showing a similar plot over different subsets of the data, usually different values of a categorical variable.

Create separate side-by-side plots for each different category of language.

```
can_lang_percent_plot
```


category
Aboriginal languages
Non−Official & Non−Aboriginal languages
Official languages
Mother tongue
(percentage of Canadian residents)
10.000
0.100
0.001
0.0001
0.0100
1.0000
100.000
Language spoken most at home
(percentage of Canadian residents)