Ling 105
Sounds of Language

Tuesday, October 22, 2024

Kevin Ryan

This week's plan

- Starting with processing & summarizing data in R
- Reading: *R for data science (2e)* parts 2–3 (online at `r4ds.hadley.nz`)
- Assignment 6 due Friday

Starting in R

- R: the most popular framework for data analysis & visualization in linguistics and many other fields
- Free for all platforms, open-source, extremely powerful; anyone can contribute packages
- Download and install R
- Then download and install RStudio
- Run RStudio, not R

RStudio

- When you open it, you'll see a prompt
- You can enter commands one-by-one
  - E.g. any math formula (using * for times and / for divide)
- But better to click on the icon in the upper-left corner to open an "R script" pane
- Here, you can enter a script/program, e.g.

```
x = 5
y = x / 2
10 * (x + y)
```

- To run it, use either the "Run" icon
  - By default, runs all lines
  - To run a subset, highlight it
- Or Command-Enter (*vel sim.*)
  - By default, runs one command at a time
  - To run several, highlight them

Packages

- Install the tidyverse package

  install.packages("tidyverse")
- Activate the package (usually the first line of a script)

  library(tidyverse)

Opening a file

- Read in data as a **data frame** with rows and columns

  x = read.csv("words12.txt", sep = "\t")

- CSV means comma-separated, but override that to specify tab-delimited using sep = "\t"
- Inspect the data frame by clicking on its name in the upper-right panel
- For more information, see words_readme.pdf on Canvas

Pipeline

- Chain commands using |>
- Start the chain with a data frame, then perform successive operations on it
- filter() keeps only rows meeting the specified condition holds
- E.g. how often is the /t/ of *winter* deleted? Get all rows in which the word is "winter":

```
x |>
    filter(Word == "winter")
```

- By default, the resulting data frame will be printed to the console
- But it's easier to save it for inspection (as "y" or any other name) by y = at the beginning or -> y at the end

filter()

- For checking equality, use ==, not =
- Can also check inequalities (>, <, >=, <=, !=)
- To check just beginning of string, filter by str_detect(Word, "^win")
- Or the end: str_detect(Word, "win$")

Summarizing by group

- Get the mean duration of each word in the corpus:
- x |>
  group_by(Word) |>
  summarize(mean_dur = mean(Duration))

Summarizing by group

- Some useful summary functions: mean(), median(), min(), max(), n(), first(), sum()
- Mean vs. median
- Get the median duration and count of each word
-
```
x |>
  group_by(Word) |>
  summarize(median_dur = median(Duration), n = n())
```

Summarizing by group and filtering

- Keep only words with frequencies of at least ten
- x |>
  group_by(Word) |>
  summarize(median_dur = median(Duration), n = n()) |>
  filter(n > 9)

Sorting

- Sort in the pipeline via arrange() (enter the column name(s))
- To sort with the highest value on top, use arrange(desc())
- E.g. what is the highest median duration of a word with at least ten tokens?
-
```
x |>
  group_by(Word) |>
  summarize(median_dur = median(Duration), n = n()) |>
  filter(n > 9)
  arrange(desc(median_dur))
```

Types vs. tokens

- How many words are in the corpus?
- Depends on whether you're counting **tokens** or **types**
- Token: every word instance, including repeats
- Type: each unique word counted only once
- For # of rows, look at data panel, or use nrow()
- To see a guide for any command, prepend ?, e.g. ?nrow
- How many word tokens end with *-nth*? What about types?

Practice

- Which speaker says *the* the most slowly on average?
- Which speaker says *um* the most number of times?
  - Use sum(Word == "um") in summarize()
- Which speaker says *um* the most **per word**?