*Summary + some interesting findings for Shiny App Creation! (At the end of notes)

Correspondence Analysis I and II

Libraries

- tidyverse
- tidytext
- janeaustenr
- FactoMineR

The janeaustenr library (Review)

- this section uses the janeaustenr library, which has a function austen_books(), which has 6 novels:
  - Emma
  - Mansfield Park
  - Northanger Abbey
  - Persuasion
  - Pride and Prejudice
  - Sense and Sensibility

  all texts of each book can be accessed by austen_books()

Intro to Correspondence Analysis (CA)

- Correspondence Analysis (CA) allows us to detect associations

Ex: Suppose we want to analyze the use of punctuation in austen_books.

punctuations → commas, semicolons, colons, quotations, apostrophes, question marks, exclamation marks, dashes

*Recall from the Regex lectures, we can get a count of a SPECIFIC character by using str_count()

To get a data frame for the symbols listed above.

austen_books (columns) → book, text

get the books and text

```
crosstable = austen_books() |>
    mutate(
        commas = str_count(text, ","),
        colons = str_count(text, ":"),
        semicolons = str_count(text, ";"),
(a)     quotes = str_count(text, "'|\""),
        apostrophes = str_count(text, "'"),
        questions = str_count(text, "\\?"),
        exclamations = str_count(text, "\\!"),
        dashes = str_count(text, "--")
    ) |>
(b) group_by(book) |>
    summarise(
        commas = sum(commas),
        colons = sum(colons),
(c)     semis = sum(semicolons),
        quotes = sum(quotes),
        apost = sum(apostrophes),
        quests = sum(questions),
        bangs = sum(exclamations),
        dashes = sum(dashes)
    )
```

("'") using single quotes is another way to make a string!

we want a count for each book

what is the above code block doing?

RECALL the austen_books() function:

austen_books() → | book | text | → (a) new columns, counts respective character in each row of text

(b) group by each unique book

(c) new table with each UNIQUE book + the totals counts

crosstable

Crosstable

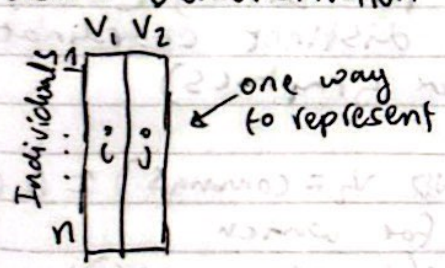
Var 1 ↓ Var 2 →

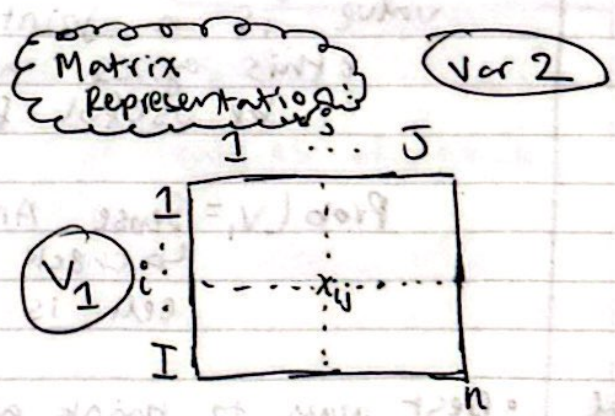
	book	commas	colons	...	dashes (9 columns)
1	Sense &...	9900	66	...	1178
2	Pride &...	9132	132	...	395
3	Mansfield...	12439	339	...	413
4	Emma	12020	174	...	3100
5	Northanger	6085	83	...	419
6	Persuasion	7025	130	...	142

What is a crosstable?

- Basically a 2-way table or contingency table (one variable on the left side [book] and the second variable on the right side [commas, colons, ..., dashes])

Visual Demonstration:





* Great for finding associations or correlations

Reformatting into a Crosstable

Step 1: Represent data frame as a matrix

```
X = as.matrix(crosstable[,-1])
rownames(X) = str_extract(crosstable$book, "\\w+")
```

only the first word

this removes the 'book' name for the column allowing us to visualize each respective variable.

Side Note: We can calculate probabilities and relative frequencies! (Helps with correlation)

Joint Probability (Relative Frequency)

Xprobs = X / sum(X) ← total count of symbols
round(Xprobs, 4)

Xprobs:

	commas	colons	...	dashes
Sense	0.0967	0.0006	...	0.0115
Pride	0.0892	0.0013	...	0.0039
Mansfield	0.1216	0.0033	...	0.0040
Emma	0.1175	0.0017	...	0.0303
Northanger	0.0595	0.0008	...	0.0041
Persuasion	0.0687	0.0013	...	0.0014

In math terms:

$f_{ij} = \frac{x_{ij}}{n}$

or

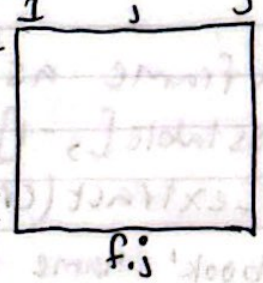
new value = $\frac{\text{current value}}{\text{sum of all counts}}$

Best way to think about each cell value is a joint probability (Var1, Var2):
- this gives a distinct coordinate (will be useful for graphics)

Prob(V_1 = Sense AND V_2 = commas) ≈ 0.0967
↳ check for which cell is ((Sense, commas));

Marginal Probability

- Best way to think about it:
  - what would the NEXT probability look like V_2 ↳ given the rows and columns



$$f_{i\cdot} = \sum_{j=1}^{J} f_{ij}$$

$$f_{\cdot j} = \sum_{i=1}^{I} f_{ij}$$

Row Margin: Sum of entries of each row, by row

$$f_{i\cdot} = \sum_{j=1}^{J} f_{ij}$$

R Coding: we can use rowSums(Matrix)

```
row_margin = rowSums(xprobs)
round(row_margin, 4)
```

Sense	Pride	...	Persuasion
0.1732	0.1606		0.1104

$P(V_1 = \text{Sense}) = 0.1732$

These proportions are **marginal probabilities**

The same for column margins:

Column Margin: Sum of entries of each column, by column

$$f_{\cdot j} = \sum_{i=1}^{I} f_{ij}$$

R Coding: we can use colSums(Matrix)

```
col_margin = colSums(Xprobs)
round(col_margin, 4)
```

commas	colons	...	dashes
0.5531	0.009		0.0552

$P(V_2 = \text{commas}) = 0.5531$

Independence Model

Probability Review:

$P(A \text{ and } B) = P(A) \cdot P(B)$ if A and B are independent

So now we recall the different probabilities we just covered:

$P(\text{Joint}) : f_{ij}$

$P(\text{Marginal}_{\text{Row}}) : f_{i.}$

$P(\text{Marginal}_{\text{col}}) : f_{.j}$

Which means:

$P(\text{Joint}) = P(M_{row}) \cdot P(M_{col})$

$f_{ij} = f_{i.} \cdot f_{.j}$

Assume V_1 is independant to V_2:

Then the table of joint probabilities should be approximately the product of the marginal probabilities

R Coding:

```
Xindep = row_margin %o% col_margin
round(Xindep, 4)
```

The %o% operator: gives the "outer" product of arrays

Note*: when you do something like 1:3 %o% 1:3 you get:

```
   [1] [2] [3]
[1]  1   2   3
[2]  2   4   6
[3]  3   6   9
```

∴ This should give us the joint probabilities (an approximate)*

* because of rounding

+ mosaicplot(x, main, sub, xlab, ylab, sort, off, dir, color, shade, margin, cex.axis, las, border, type)

With our array, we can make many plots (the Correspondence Analysis Notes uses a mosaicplot+)

To make a mosaic plot:

↳ you need some sort of CONTINGENCY table

Arguments:

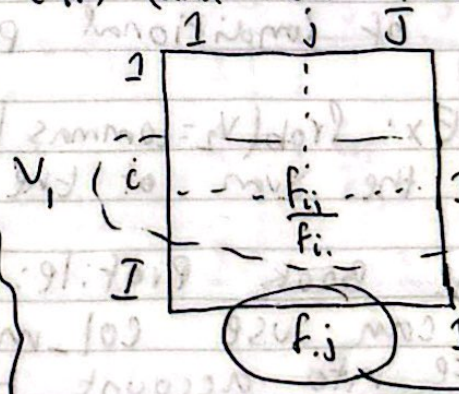
- x → contingency table
- main → character string (title)
- sub → like a subtitle; character string
- xlab, ylab → just like labs() in ggplot, x label and y label
- las → numeric (style of axis labels)
- border → color of borders of cells

Row Analysis

→ Recall our table of probabilities: Xprobs

In order to conduct a Row Analysis, we divide each entry (f_{ij}) by the ROW MARGIN ($f_{i.}$) (aka our marginal probability)

$$\frac{f_{ij}}{f_{i.}} = f_{.j}$$



This is called a "Row Profile"
↳ a conditional distribution

Mean row profile
↳ the distribution profile for ALL individuals

CA takes these row profiles and compares against the mean profile

How do we get "Row Profiles" in R?

we can use the sweep() function

Sweep():
- X → an array (matrix allowed)
- MARGIN → numeric; vector of indices, giving correspondence of x to STAT
  or char vector of dimension names
- STATS → summary statistic
- FUN → "[function you want to use]"
  - Ex: "-" → subtract
    "+" → add
    "/" → divide

~~Create the row~~

```
row_profiles = sweep(Xprobs, MARGIN=1, STATS=
                     row_margin, FUN="/")
round(row_profiles, 4)  → rounds to 4 decimal places
```

Our new table is basically a table of conditional probabilities

Ex: $Prob(V_2 = commas \mid V_1 = Sense) = 0.5585$

* All the sum of the rows are 1

Average Book Profile: ~~instead~~

we can use col_margin to take into account book profiles

* you can use round() to round entries of a vector, table (complex vectors) to some decimal places

```
Rows = rbind(row_profiles, average = col_margin)
round(Rows, 4)

mosaicplot(
    t(Rows),
    las = 1,
    border = NA,
    col = rainbow(ncol(Rows)),
    main = "Row Profiles")
```

→ Constructs a Mosaic plot based on our new Rows table

Based on the mosaicplot, we can make several observations about our Row Analysis

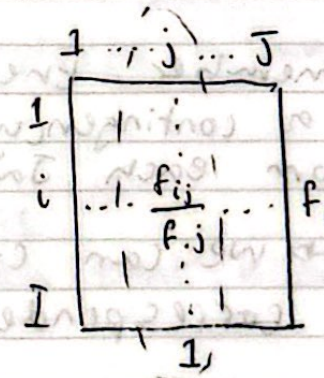
Ex: - commas are the most used punctuation symbol (evenly distributed across all books)

Column Analysis

- Very similar to Row Analysis
  - ↳ we use col_margin

Analysis by Column:

$$\frac{f_{ij}}{f_{.j}} = f_{i.}$$



Mean Column Profile

Once again CA compares column profile to the mean profile

Column profile j
↳ Conditional Distribution

```
col_profiles = sweep (Xprobs, MARGIN=2, STATS = col_margin,
        FUN = "/")
round (col_profiles, 4)
```

Average Symbol Profile: Take into account the average symbol profile (row_margin)

```
Cols = cbind(col_profiles, average = row_margin)
round (Cols, 4)

mosaicplot(
    t(Cols),
    las = 1,
    border = NA,
    col = rainbow(nrow(Cols)),
    main = "Column Profiles")
```

Example observation based on the plot:
- Mansfield has the largest proportion of colons

Correspondence Analysis Map

- Remember the crosstable which is a contingency table of punctuations from each Jane Austen novel
  - ↳ we can create a map for our correspondence analysis

```
X = as.matrix(crosstable[,-1])
rownames(X) = str_extract(crosstable$book, "\\w+")
```

X → new matrix representation of crosstable

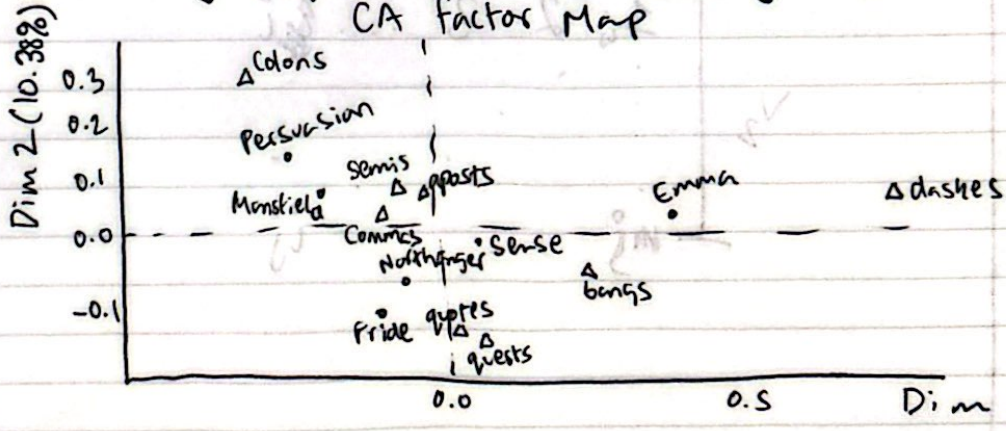
Simultaneous Rep of Rows and Columns

- CA() → from "FactoMineR"
  - allows us to perform that Correspondence Analysis using a matrix

Ex:

austen_ca1 = CA(X)

*by default, creates a Scatterplot to visually represent the categories



- We can disable the default graph with the argument graph = FALSE
  - we can use ggplot() to make custom CA maps

Summary

- Correspondence Analysis
  - a type of analysis that allows us to detect associations

Overall Process

Data Frame → CrossTable (Contingency Table) → Statistical Analysis Tool

Some Analyses we can do:

- Row Analysis
- Column Analysis
- Creating a Map

★ Contingency Tables are a data frame mapped with 2 variables: