Andrew Kerr
California Polytechnic State University, San Luis Obispo
05/22/2025

# Frequent Itemset Mining with tidyclust in R

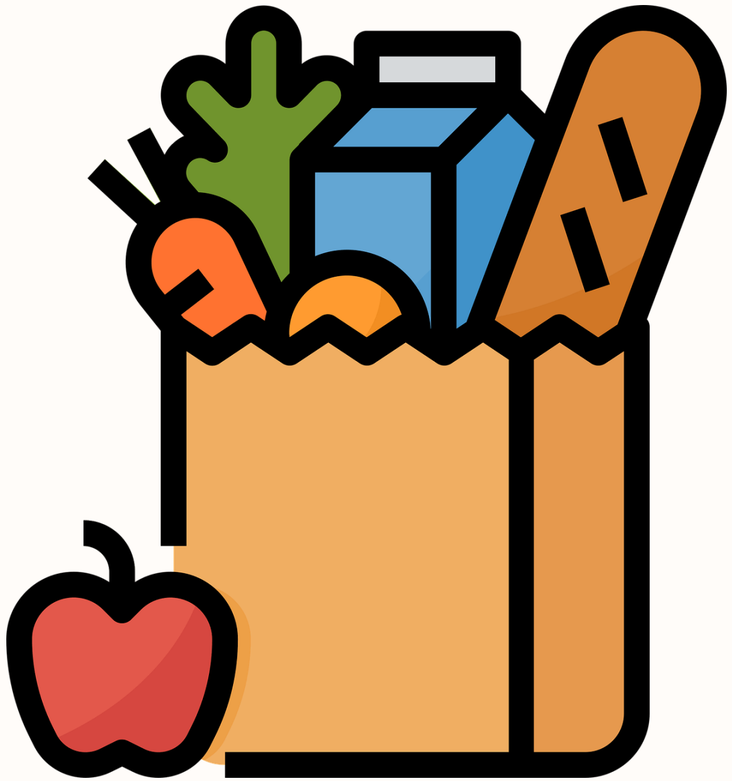
Advisor: Dr. Bodwin
Committee: Dr. Glanz and Dr. Dekhtyar

1. What is Frequent Itemset Mining
2. Explaining tidyclust
3. Clustering Frequent Itemsets
4. Predicting with Frequent Itemsets
5. Future Directions

# What is Frequent Itemset Mining?

# Finding Co-occurrences

- Frequent Itemset Mining (FIM) helps us find items that often appear together in a dataset.
- For example, FIM may discover that customers who buy coffee also frequently buy milk.
- Key Terms
  - Itemset: A **collection of items**.
  - Frequent Itemset: A collection of items that are **commonly found together**.
  - Minimum Support: The **cutoff point** that determines if an itemset is frequent.



| beer | milk | bread | diapers | eggs |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |

# Uses for Frequent Itemset Mining

## Market Basket Analysis

Understanding customer buying habits (e.g., products frequently purchased together).

## Medical Diagnosis

Identifying co-occurring symptoms or conditions.

## Fraud Detection

Detecting unusual combinations of transactions or activities.

Andrew Kerr

California Polytechnic State University, San Luis Obispo

05/22/2025

# A Refresher on tidyclust.

```
fi_spec <- freq_itemsets(
  min_support = 0.05,
  mining_method = “apriori”,
  ) |>
   set_engine("arules")

fi_spec_fit <- fi_spec |>
  fit(~., data = groceries)

fi_spec_fit |>
  predict(new_groceries)
```

# tidyclust Workflow

- tidyclust is a set of tools in R that helps find patterns in data without pre-defined labels (unsupervised learning).
- The goal of tidyclust is to establish a consistent and reproducible workflow.
- A user can:
  - Specify a model
  - Fit a model
  - Predict with a model

  all using a standardized syntax.

Andrew Kerr

California Polytechnic State University, San Luis Obispo

05/22/2025

# Clustering Frequent Itemsets.

# Clustering

- Group similar data points (rows) together.
- Finds natural groupings based on data characteristics.

| sex | island | bill length | bill depth | body mass |
|---|---|---|---|---|
| male | Torgersen | 39.1 | 18.7 | 3750 |
| female | Torgersen | 39.5 | 17.4 | 3800 |
| female | Torgersen | 40.3 | 18 | 3250 |
| female | Torgersen | 36.7 | 19.3 | 3450 |
| male | Torgersen | 36.8 | 20.6 | 3650 |

# Clustering with FIM

- Groups similar items (columns) together.
- Groups items based on co-occurrence in transactions.

| beer | milk | bread | diapers | eggs |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |

# Finding Frequent Itemsets

| beer | milk | bread | diapers | eggs |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |

$$\text{Support}(X) = \frac{\text{Number of transactions containing itemset X}}{\text{Total number of transactions}}$$

Minimum Support: 0.5

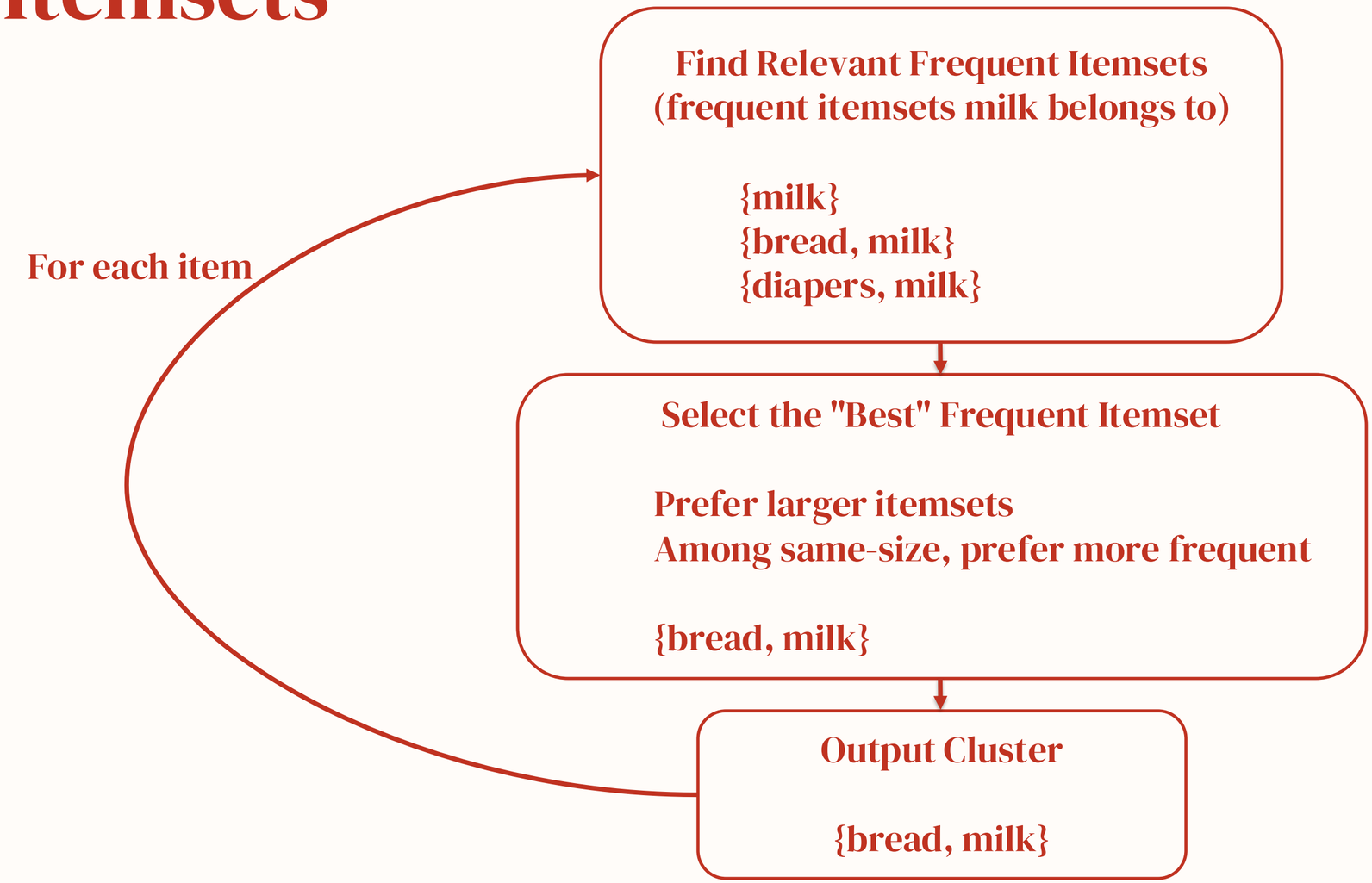
**Apriori Principle**: If an itemset is frequent, then so are all its subsets.

| item | support |
|---|---|
| beer | 0.6 |
| milk | 0.8 |
| bread | 0.8 |
| diapers | 0.8 |
| eggs | 0.2 |

| itemset | support |
|---|---|
| {beer, bread} | 0.4 |
| {beer, diapers} | 0.6 |
| {beer, milk} | 0.4 |
| {bread, diapers} | 0.6 |
| {bread, milk} | 0.6 |
| {diapers, milk} | 0.6 |

| itemset | support |
|---|---|
| {bread, diapers, milk} | 0.4 |

# Clustering Frequent Itemsets




**Handling Outliers**

Items not in any frequent itemset are outliers.

Andrew Kerr

California Polytechnic State University, San Luis Obispo

05/22/2025

# Predicting with Frequent Itemsets.

| beer | milk | bread | diapers | eggs |
|---|---|---|---|---|
| 1 | NA | 0 | 1 | 0 |

| itemset | support |
|---|---|
| milk | 0.8 |
| bread | 0.8 |
| diapers | 0.8 |
| beer | 0.6 |
| {beer, diapers} | 0.6 |
| {bread, diapers} | 0.6 |
| {bread, milk} | 0.6 |
| {diapers, milk} | 0.6 |

| itemset | support | confidence |
|---|---|---|
| {diapers, milk} | 0.6 | 0.75 |

| milk |
|---|
| 0.75 |

# Predicting with Frequent Itemsets

Given a partial set of items, for each missing item:

1. Find Relevant Frequent Itemsets: Find frequent itemsets that contain the missing item and at least one observed item.
2. Calculate Confidence: The likelihood of the missing item appearing, given the observed items.

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X \text{ or } Y)}{\text{Support}(\text{observed items in } X)}$$

3. Predict: Average these likelihoods.

# k-means Output

| .pred_cluster |
|---|
| Cluster_1 |
| Cluster_1 |
| Cluster_2 |
| Cluster_3 |
| Cluster_2 |

# FIM Output

| .pred_cluster |
|---|
| <df [5 x 3]> |
| <df [5 x 3]> |
| <df [5 x 3]> |

| item | .obs_item | .pred_item |
|---|---|---|
| beer | 0 | NA |
| milk | NA | 1 (0.75) |
| bread | 1 | NA |
| diapers | 0 | NA |
| eggs | 0 | NA |

# Future Additions.

# Future Directions

## Parameter Tuning

Automate minimum support tuning with data-driven ranges.

## Cross Validation

Implement item-stratified cross-validation for robust model evaluation.

## Enhance Prediction

Test different confidence weight metrics.

# Tuning Minimum Support

- The optimal minimum support value varies depending on the characteristics of the dataset.

- I propose

  1. Calculate the **mean support** of all 1-itemsets:

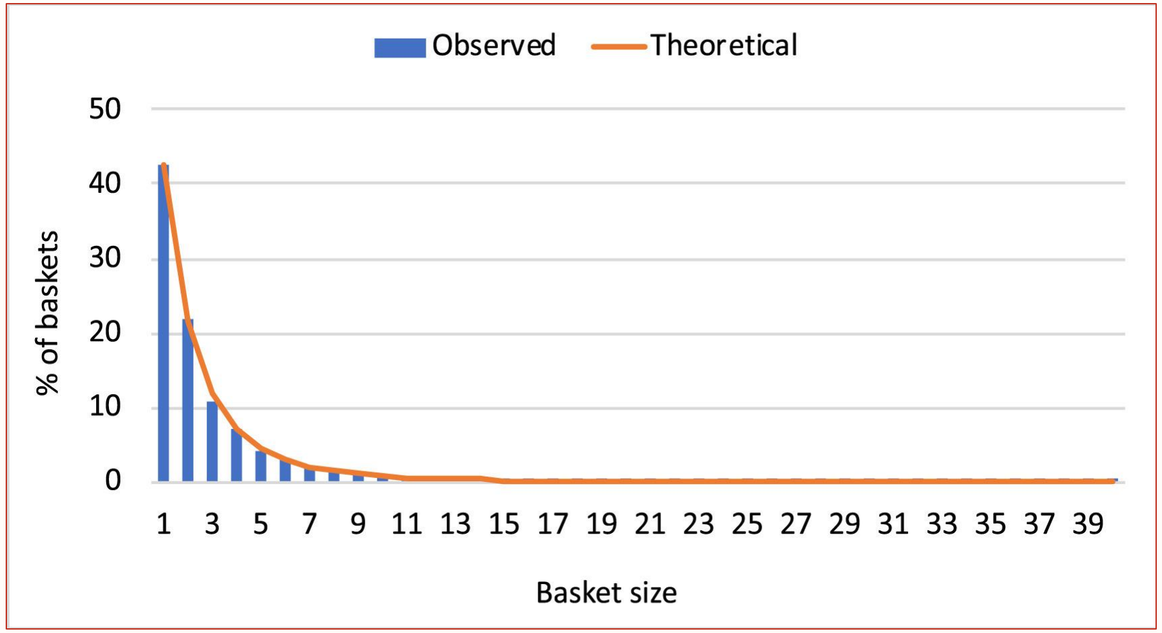
$$\mu = \frac{1}{n}\sum^{n} support(\{item_i\})$$

  2. Create a confidence-interval-like **range around $\mu$** using the standard deviation $\sigma$:

$$\left[\mu - \frac{\sigma}{2}, \mu + \sigma\right]$$

  clipping the bounds at [0, 1] to ensure valid support values.

  The range is **asymmetric** since transactional datasets tend to have a **right skewed** support distribution.




From *Fundamental basket size patterns and their relation to retailer performance*, Martin et. al. 2009

# Putting Everything Together.

# Functions!

```
freq_itemsets():
     .freq_itemsets_fit_arules()

fit():
     extract_cluster_assignment.itemsets()
     item_assignment_tibble_w_outliers()

predict():
     itemsets_predict_helper()
     .freq_itemsets_predict_raw_arules()
     .freq_itemsets_predict_arules()
     extract_predictions()
     augment_itemset_predict()

Misc.():
     extract_fit_summary_items()
     tunable.freq_itemsets()
     random_na_with_truth()
```

# Thank you!

(Especially Dr. Bodwin!)