==================================================================

**Below: Second round of themes (after initial use on the survey data by both raters and discussion of rare or correlated codes)**

- **How they differ**
  - SEM-KNOW: PTM names have more semantic knowledge
    - Model heritage (fine-tuning from model X)
    - Model parameters
    - Model version (architecture info)
    - (variation among PTMs) Some semantic knowledge is especially important (though varying opinion on which knowledge matters)
  - DIFF-KINDS: Different kinds of names ("archetypal"/how it works and "aesthetic"/cute/fun)
  - MORE-VARIATION | VARYING-NAMES: PTM names have more variations in how people write them
- **Why they differ**
  - FORK: PTMs have more forking / PTMs and traditional packages evolve and get adapted differently / PTM variations are "low depth, high width"
  - REUSE: PTM users care more about how it is implemented. Traditional package users care more about how to use it. PTM reuse requires more adaptation compared to traditional package reuse. (So users need more details). One example of this is the former code "CONFIG":
    - *CONFIG*: PTMs are more single-purpose / less configurable so the name encodes the built-in configuration to facilitate PTM selection
  - VERSION: Meaning of versioning is different (PTM-architecture; Traditional-backwards compatibility) so vX gets put into PTM names
  - README: Traditional packages put semantic knowledge elsewhere (not in name). PTM names replace documentation.
    - The ways to communicate package details are different
    - (JD: Possibly this is an optimization over reading the docs)
    - Current PTM naming conventions are due to historical accident (formerly no model cards for the metadata people want)
  - NO-STD: PTM naming conventions are unsettled, diverse, "wild west", etc
    - There is no standard (and it may be unclear what parts are important to standardize)
    - PTM naming conventions tend to follow foundational models

  - COMMS: PTM names are more verbose / have more acronyms / hard to read…because they are trying to communicate many fields concisely across many fields and possibly with reference to external literature that defines acronyms
  - OTHERS (not coded)
    - RTP: Different kinds of workers are doing the naming and that influences what they value
      - Research-to-practice pipeline
      - PTM creators are more researchers rather than engineers
    - BUZZ: Hype cycle in PTMs

==============================================================================

**Below: First round of themes (after discussion by both raters)**

- **How they differ**
  - SEM-KNOW: PTM names have more semantic knowledge
    - Model heritage (fine-tuning from model X)
    - Model parameters
    - Model version (architecture info)
    - (variation among PTMs) Some semantic knowledge is especially important (though varying opinion on which knowledge matters)
  - DIFF-KINDS: Different kinds of names (“archetypal”/how it works and “aesthetic”/cute/fun)
  - MORE-VARIATION: PTM names have more variations
- **Why they differ**
  - FORK: PTMs have more forking / PTMs and traditional packages evolve and get adapted differently / PTM variations are “low depth, high width”
  - CONFIG: PTMs are more single-purpose / less configurable so the name encodes the built-in configuration
  - REUSE: PTM users care more about how it is implemented. Traditional package users care more about how to use it. PTM reuse requires more adaptation compared to traditional package reuse. (So users need more details)
  - VERSION: Meaning of versioning is different (PTM-architecture; Traditional-backwards compatibility) so vX gets put into PTM names
  - README: Traditional packages put semantic knowledge elsewhere (not in name). PTM names replace documentation.
    - The ways to communicate package details are different
    - (JD: Possibly this is an optimization over reading the docs)

        - Current PTM naming conventions are due to historical accident (formerly no model cards for the metadata people want)
    - NO-STD: PTM naming conventions are unsettled, diverse, “wild west”, etc
        - There is no standard (and it may be unclear what parts are important to standardize)
        - PTM naming conventions tend to follow foundational models
    - RTP: Different kinds of workers are doing the naming and that influences what they value
        - Research-to-practice pipeline
        - PTM creators are more researchers rather than engineers
    - BUZZ: Hype cycle in PTMs
    - COMMS: PTM names are more verbose / have more acronyms / hard to read…because they are trying to communicate many fields concisely across many fields and possibly with reference to external literature that defines acronyms