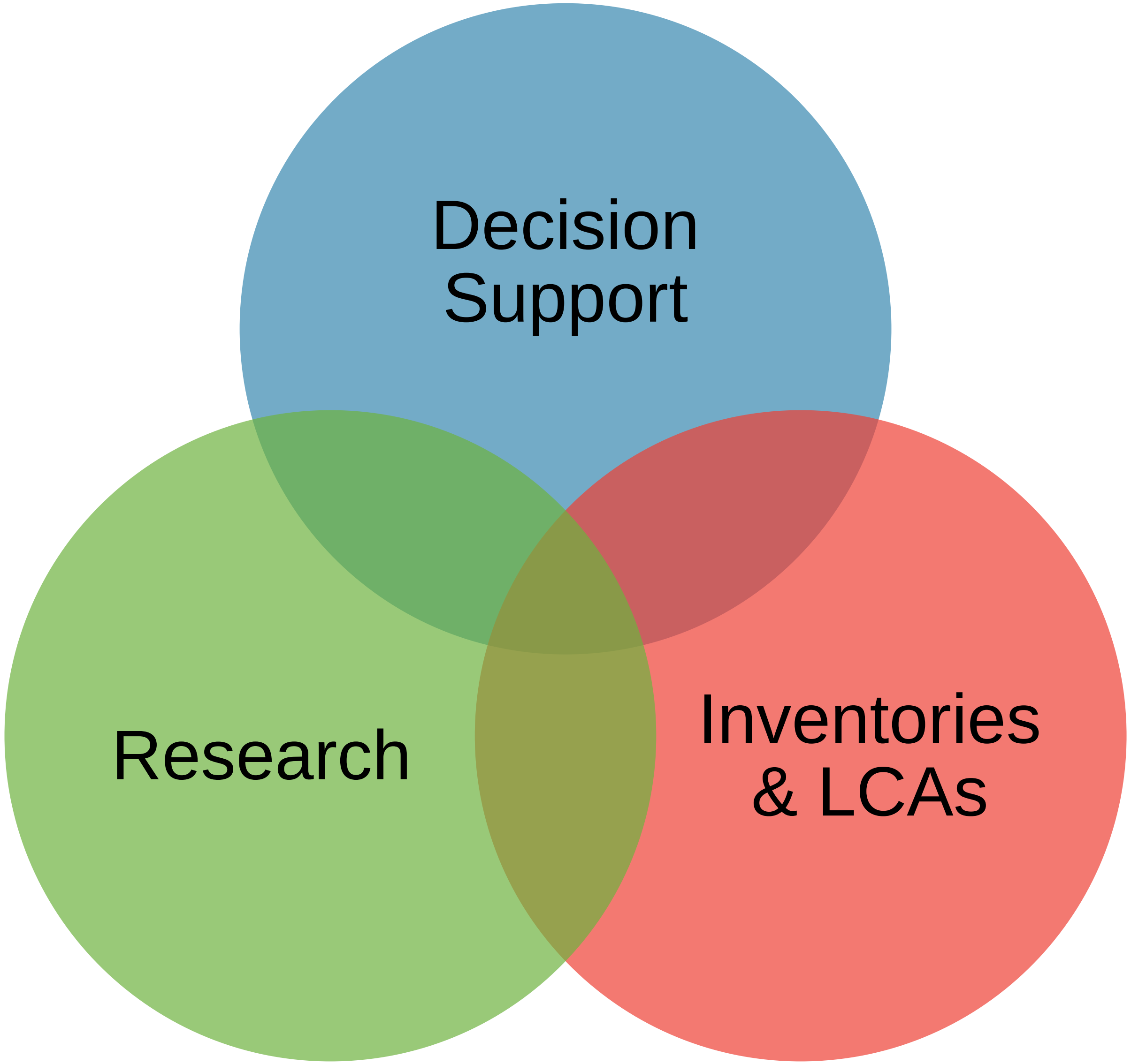
Modeling Research and Application to Support the Future of Dairy Production

Kristan Reed, PhD
Assistant Professor of Dairy Cattle Nutrition
NEAFA Sesquicentennial Fellow

The Role of Models

Models are often developed to serve one of these three purposes.

Occasionally models can bridge the gap from a research application or inventory into a decision support tool as well.



Current Research

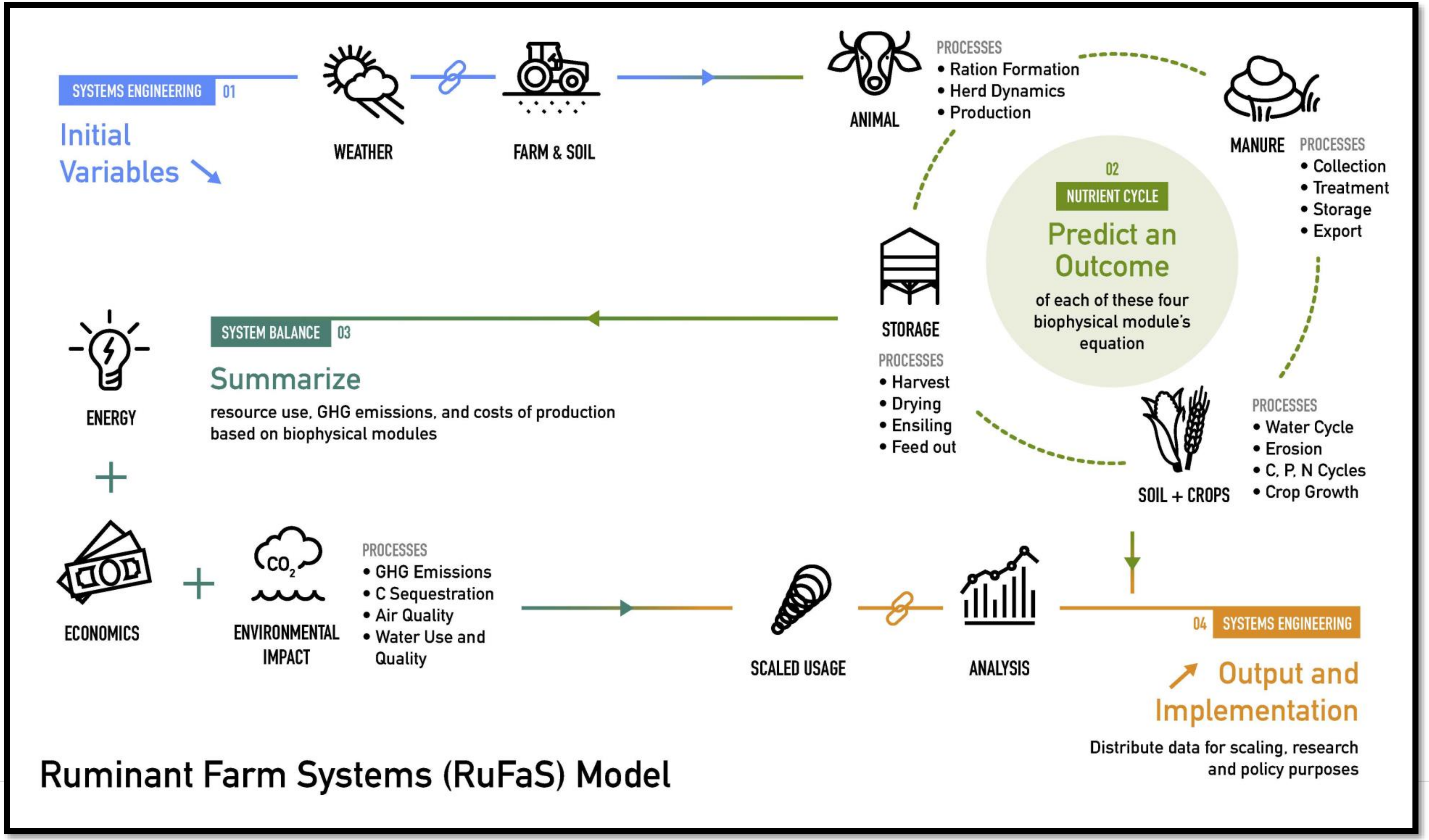
The RuFaS Model

- Progress in Model Development & Evaluation
- Future Directions

Managing Feed Variability

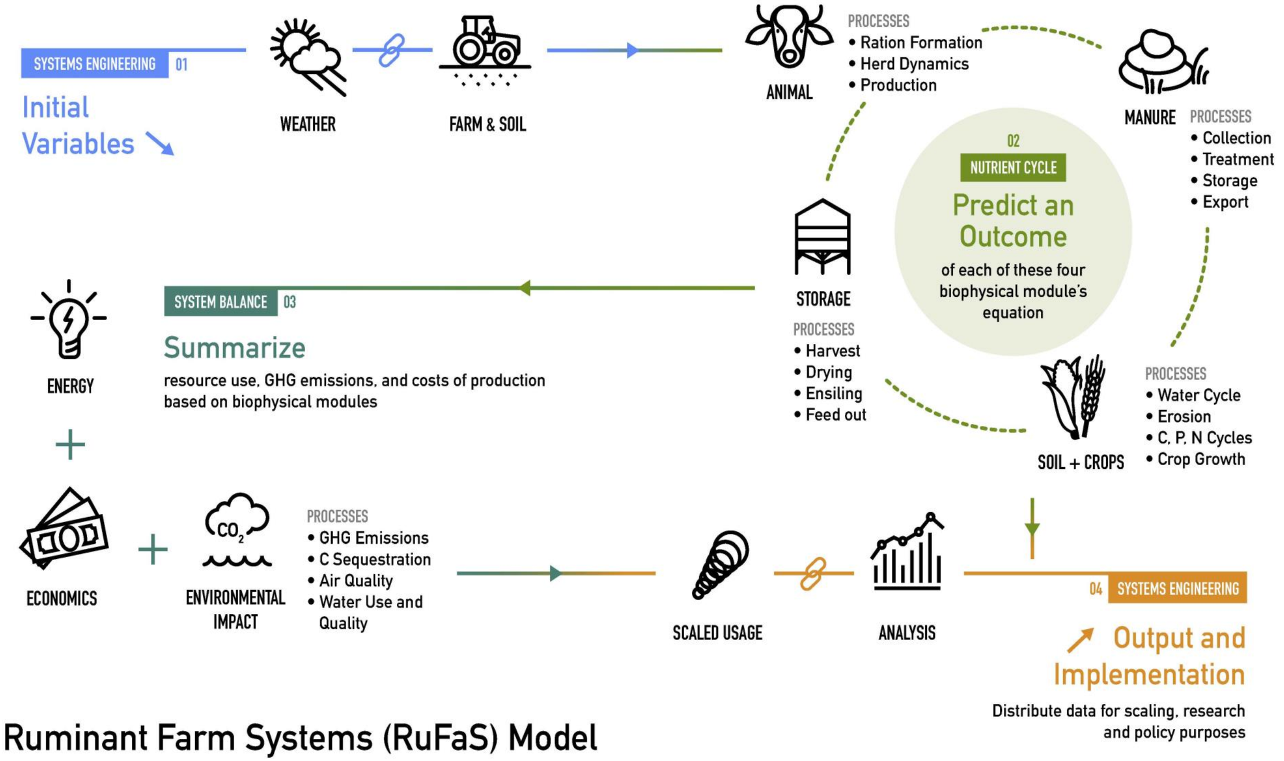
- Quantifying and Identifying Sources of Feed Variation
- Improving Efficiency through Management

What is RuFaS?



The RuFaS Vision

To ***support research and sustainable decision-making*** in ruminant animal production through ***a state-of-art, open-source modeling environment*** that is continuously adapting as technology and scientific knowledge advance.



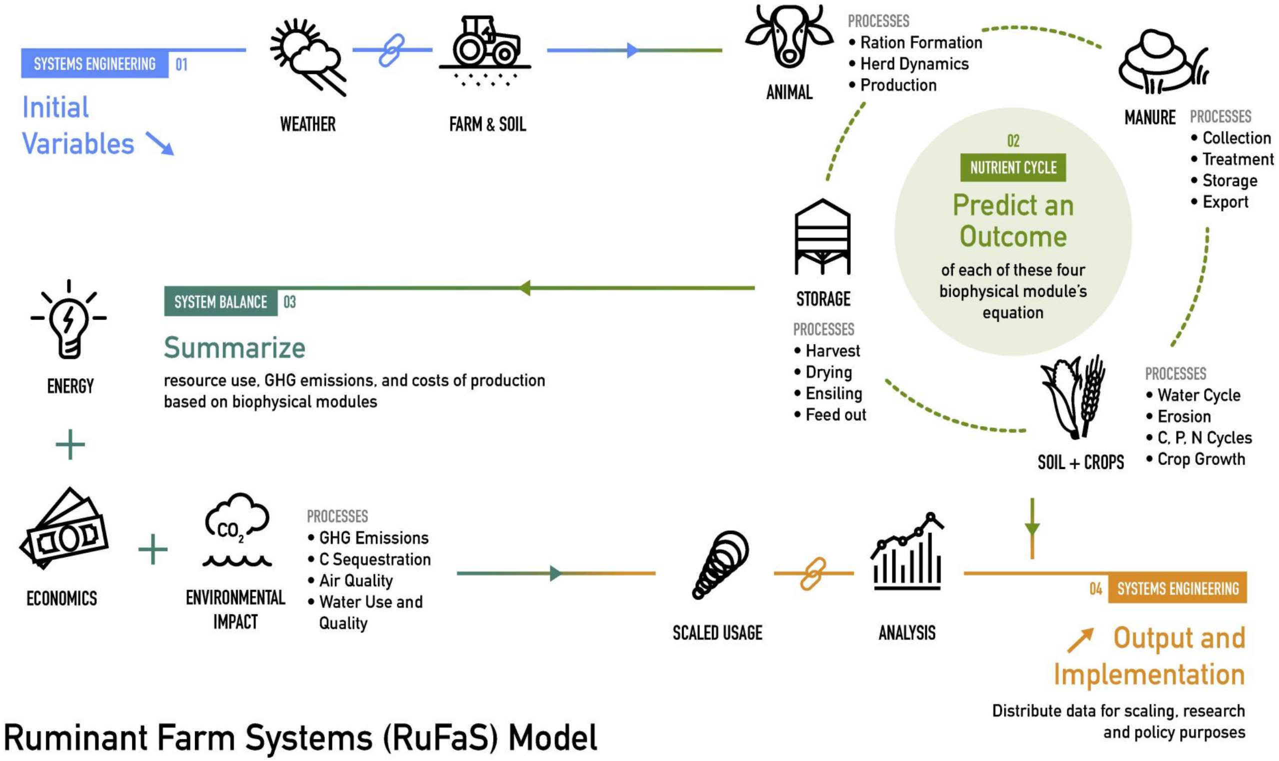
Ruminant Farm Systems (RuFaS) Model

The RuFaS Mission

To ***build an integrated, whole-farm model*** that simulates milk, meat, and crop production, greenhouse gas emissions, water quality impacts, soil health, and other ***sustainability outcomes*** of ruminant farms.

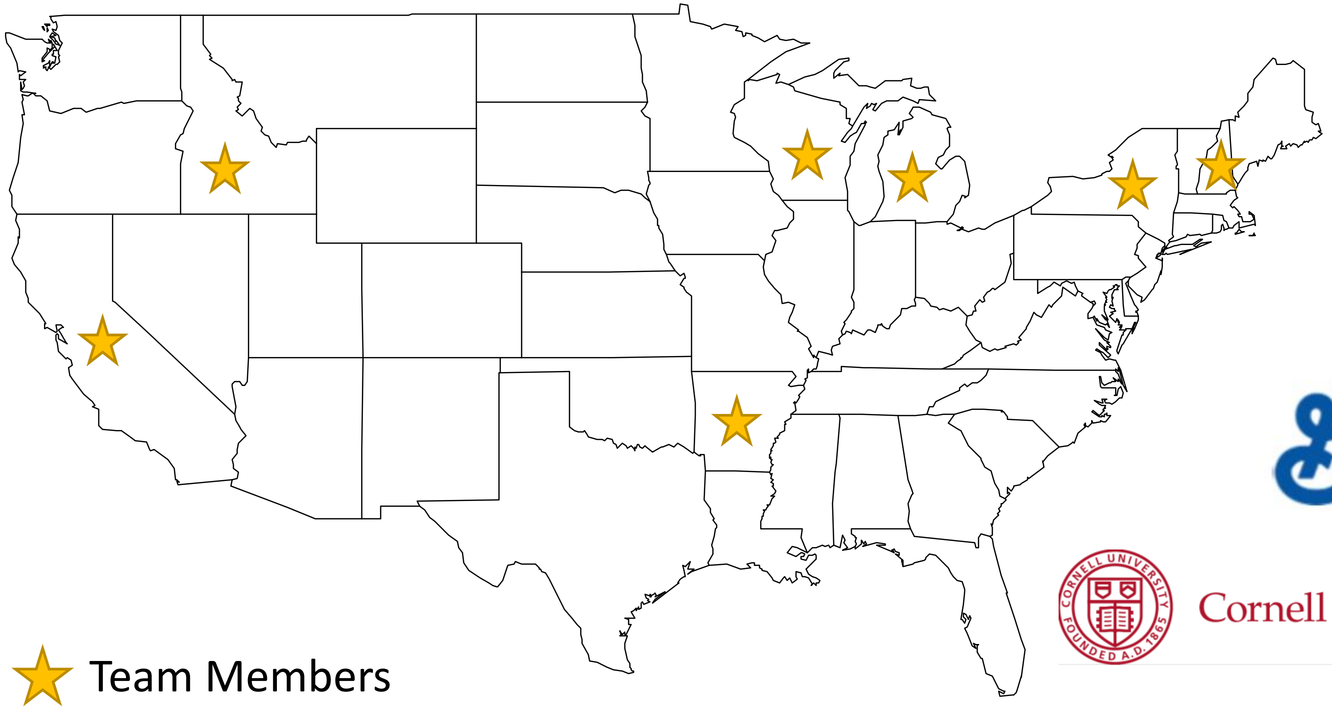
We strive to achieve the ***highest standards for prediction accuracy***, ***code structure*** and clarity, ***documentation***, and ***accessibility***.

Through ***continuous learning*** and improvement of our methods and algorithms, we are ***creating an open and inclusive platform*** for scientific collaboration.



Ruminant Farm Systems (RuFaS) Model

RuFaS Team



★ Team Members

General Mills
Making Food People Love

Cornell University

USDA

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

INNOVATION CENTER FOR U.S. DAIRY
HEALTHY PEOPLE • HEALTHY PRODUCTS • HEALTHY PLANET

The Nature Conservancy

UC DAVIS
UNIVERSITY OF CALIFORNIA

NEAFA
Northeast Agribusiness & Feed Alliance
Advocate · Educate · Collaborate

UNIVERSITY OF ARKANSAS

UNIVERSITY OF SOUTH DAKOTA

WWF

ANIMAL

MANURE

SOIL + CROPS

ENERGY

ECONOMICS

RuFaS Evolution

2017

Formalized vision and established principles for RuFaS development

2018

First lines of code written

Submitted first federal proposal

Expanded collaborators to include Cornell

2019

First paper published

Initiated documentation and version control methods

Submitted first successful Federal proposal

2020

Formalized Industry Advisory Council

Connected nutrient cycle in biophysical model

2021

Published 2 manuscripts on Animal Module

Hired first Professional Software Engineer

2022

Initiated contract with FARM-ES

2 additional manuscripts on Animal Module

2023

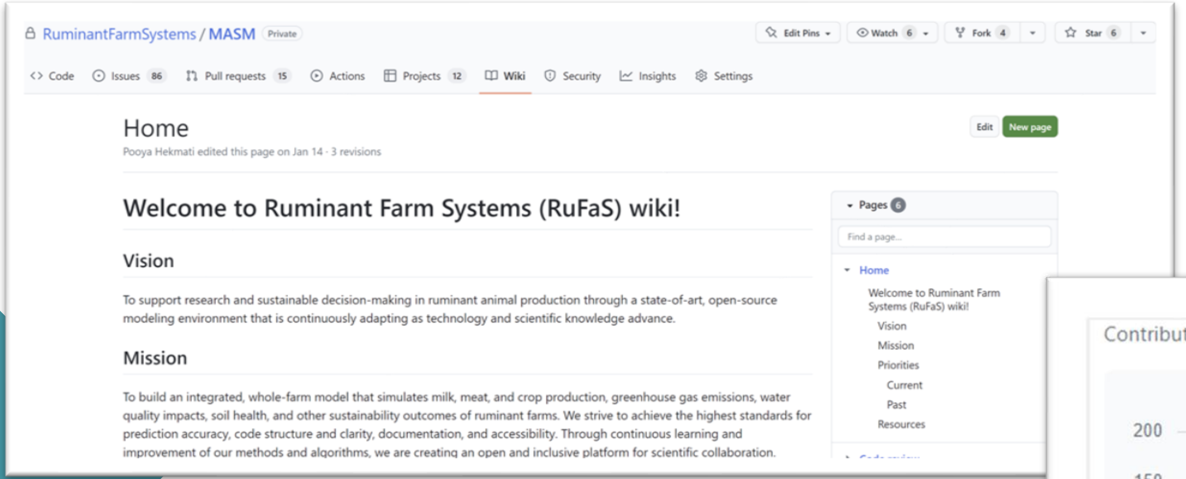
Announced Positions for 2 permanent USDA-ARS scientist positions for RuFaS

Preparing for V1 completion

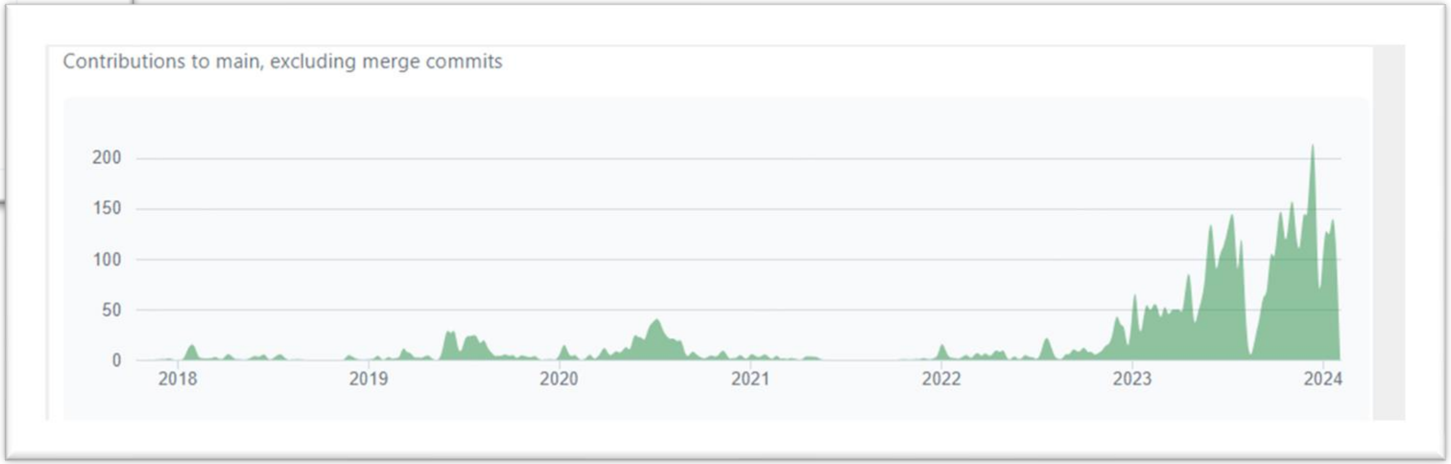
Model Progress

Working towards Version 1 Release

Preparing GitHub Repo and Documentation



Developing better workflow for consistent progress



CODE + DOCUMENTATION

CODEBASE: PROGRESS BY THE NUMBERS

84K

LINES OF CODE

Since November 2022 we have more than doubled our codebase

4.4K

UNIT TESTS

Unit tests ensure that each step in the model is functioning as expected and provides a warning when it fails

5K

TYPE ANNOTATIONS

Typing ensures the model and users know what for a variable is expected to take (e.g. number, word, category)

94%

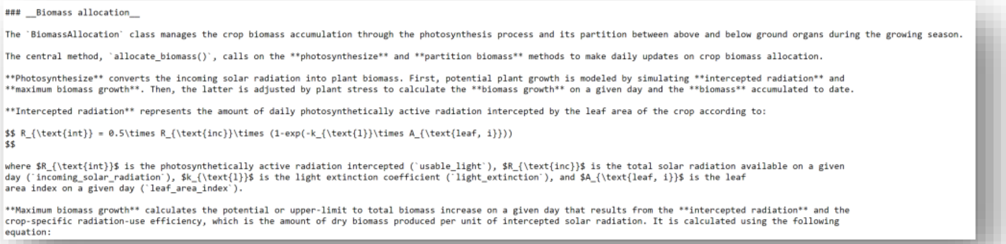
CODE COVERAGE

The percent of code that is covered by unit tests has increased dramatically since November 2022

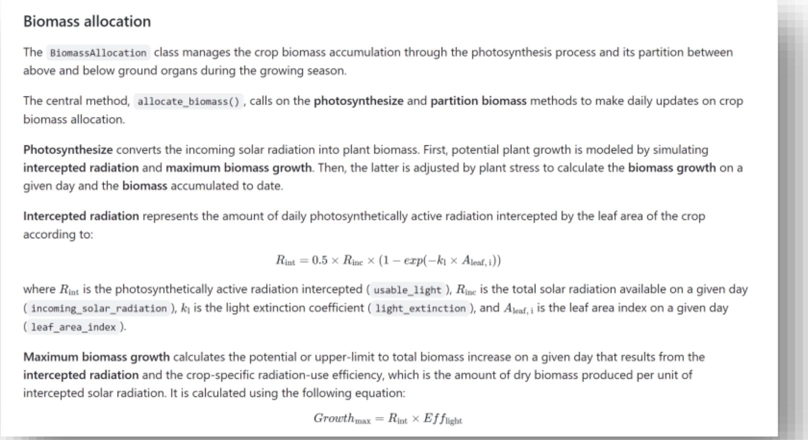
Progress in Model Documentation

Scientific Documentation

Coded in LateX or Rmarkdown – stored on designated repo folder and organized by module



```
### __Biomass allocation__

The `BiomassAllocation` class manages the crop biomass accumulation through the photosynthesis process and its partition between above and below ground organs during the growing season.

The central method, `allocate_biomass()`, calls on the **photosynthesize** and **partition biomass** methods to make daily updates on crop biomass allocation.

**Photosynthesize** converts the incoming solar radiation into plant biomass. First, potential plant growth is modeled by simulating **intercepted radiation** and
**maximum biomass growth**. Then, the latter is adjusted by plant stress to calculate the **biomass growth** on a given day and the **biomass** accumulated to date.

**Intercepted radiation** represents the amount of daily photosynthetically active radiation intercepted by the leaf area of the crop according to:

$$ R_{\text{int}} = 0.5\times R_{\text{inc}}\times (1-exp(-k_{\text{l}}\times A_{\text{leaf, i}}))
$$

where $R_{\text{int}}$ is the photosynthetically active radiation intercepted (`usable_light`), $R_{\text{inc}}$ is the total solar radiation available on a given
day (`incoming_solar_radiation`), $k_{\text{l}}$ is the light extinction coefficient (`light_extinction`), and $A_{\text{leaf, i}}$ is the leaf
area index on a given day (`leaf_area_index`).

**Maximum biomass growth** calculates the potential or upper-limit to total biomass increase on a given day that results from the **intercepted radiation** and the
crop-specific radiation-use efficiency, which is the amount of dry biomass produced per unit of intercepted solar radiation. It is calculated using the following
equation:
```



Biomass allocation

The `BiomassAllocation` class manages the crop biomass accumulation through the photosynthesis process and its partition between above and below ground organs during the growing season.

The central method, `allocate_biomass()`, calls on the **photosynthesize** and **partition biomass** methods to make daily updates on crop biomass allocation.

Photosynthesize converts the incoming solar radiation into plant biomass. First, potential plant growth is modeled by simulating **intercepted radiation** and **maximum biomass growth**. Then, the latter is adjusted by plant stress to calculate the **biomass growth** on a given day and the **biomass** accumulated to date.

Intercepted radiation represents the amount of daily photosynthetically active radiation intercepted by the leaf area of the crop according to:

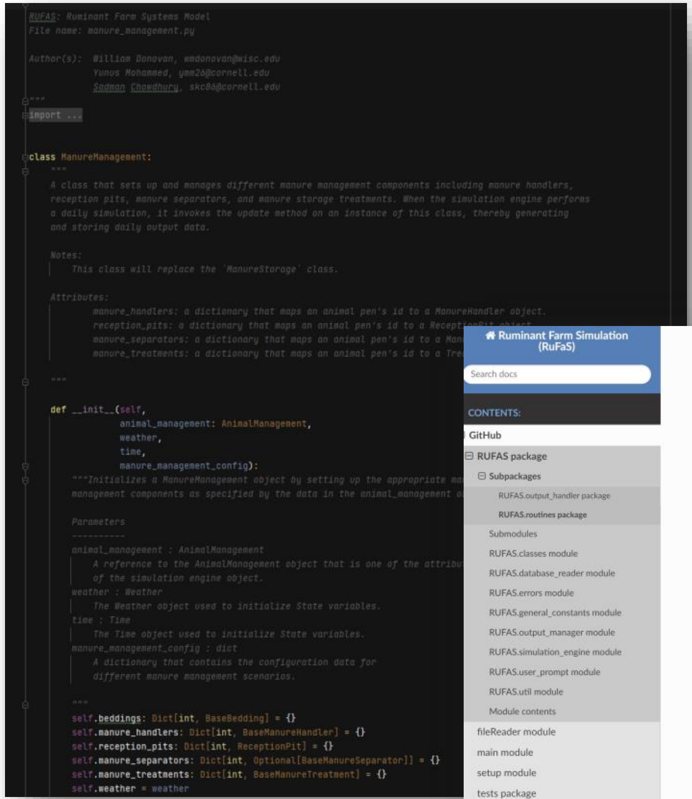
$$R_{int} = 0.5 \times R_{inc} \times (1 - exp(-k_l \times A_{leaf,i}))$$

where R_{int} is the photosynthetically active radiation intercepted (`usable_light`), R_{inc} is the total solar radiation available on a given day (`incoming_solar_radiation`), k_l is the light extinction coefficient (`light_extinction`), and $A_{leaf,i}$ is the leaf area index on a given day (`leaf_area_index`).

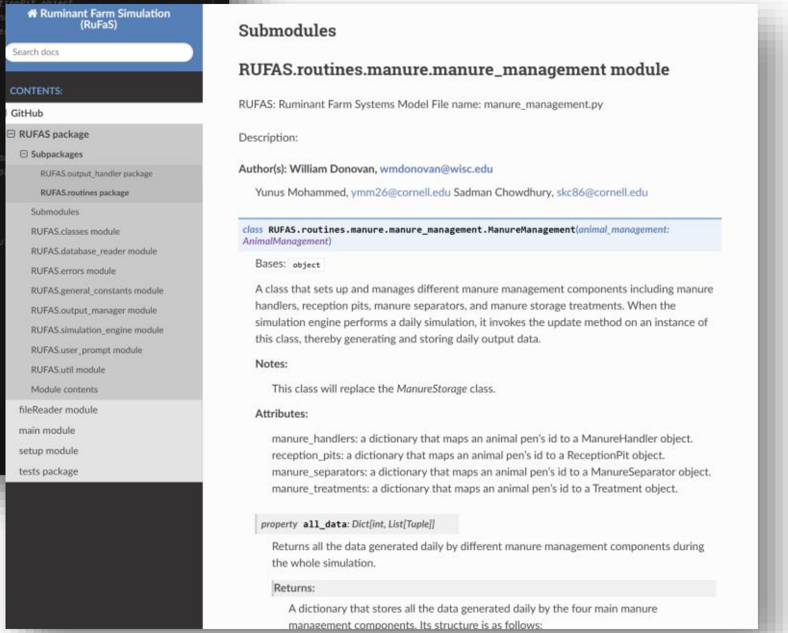
Maximum biomass growth calculates the potential or upper-limit to total biomass increase on a given day that results from the **intercepted radiation** and the crop-specific radiation-use efficiency, which is the amount of dry biomass produced per unit of intercepted solar radiation. It is calculated using the following equation:

$$Growth_{max} = R_{int} \times Eff_{light}$$

In-line Documentation of Code



```
RUFAS: Ruminant Farm Systems Model
File name: manure_management.py

Author(s):  William Donovan, wmdonovan@wisc.edu
            Yunus Mohammed, ymm26@cornell.edu
            Sadman Chowdhury, skc86@cornell.edu
"""
import ...

class ManureManagement:
    """
    A class that sets up and manages different manure management components including manure handlers,
    reception pits, manure separators, and manure storage treatments. When the simulation engine performs
    a daily simulation, it invokes the update method on an instance of this class, thereby generating
    and storing daily output data.

    Notes:
        This class will replace the `ManureStorage` class.

    Attributes:
        manure_handlers: a dictionary that maps an animal pen's id to a ManureHandler object.
        reception_pits: a dictionary that maps an animal pen's id to a ReceptionPit object.
        manure_separators: a dictionary that maps an animal pen's id to a Man
        manure_treatments: a dictionary that maps an animal pen's id to a Tre
    """

    def __init__(self,
                 animal_management: AnimalManagement,
                 weather,
                 time,
                 manure_management_config):
        """Initializes a ManureManagement object by setting up the appropriate ma
        management components as specified by the data in the animal_management o

        Parameters
        ----------
        animal_management : AnimalManagement
            A reference to the AnimalManagement object that is one of the attribu
            of the simulation engine object.
        weather : Weather
            The Weather object used to initialize State variables.
        time : Time
            The Time object used to initialize State variables.
        manure_management_config : dict
            A dictionary that contains the configuration data for
            different manure management scenarios.
        """
        self.beddings: Dict[int, BaseBedding] = {}
        self.manure_handlers: Dict[int, BaseManureHandler] = {}
        self.reception_pits: Dict[int, ReceptionPit] = {}
        self.manure_separators: Dict[int, Optional[BaseManureSeparator]] = {}
        self.manure_treatments: Dict[int, BaseManureTreatment] = {}
        self.weather = weather
```



Submodules

RUFAS.routines.manure.manure_management module

RUFAS: Ruminant Farm Systems Model File name: manure_management.py

Description:

Author(s): William Donovan, wmdonovan@wisc.edu

Yunus Mohammed, ymm26@cornell.edu Sadman Chowdhury, skc86@cornell.edu

class **RUFAS.routines.manure.manure_management.ManureManagement**(*animal_management: AnimalManagement*)

Bases: `object`

A class that sets up and manages different manure management components including manure handlers, reception pits, manure separators, and manure storage treatments. When the simulation engine performs a daily simulation, it invokes the update method on an instance of this class, thereby generating and storing daily output data.

Notes:

This class will replace the *ManureStorage* class.

Attributes:

manure_handlers: a dictionary that maps an animal pen's id to a ManureHandler object.
reception_pits: a dictionary that maps an animal pen's id to a ReceptionPit object.
manure_separators: a dictionary that maps an animal pen's id to a ManureSeparator object.
manure_treatments: a dictionary that maps an animal pen's id to a Treatment object.

property **all_data**: *Dict[int, List[Tuple]]*

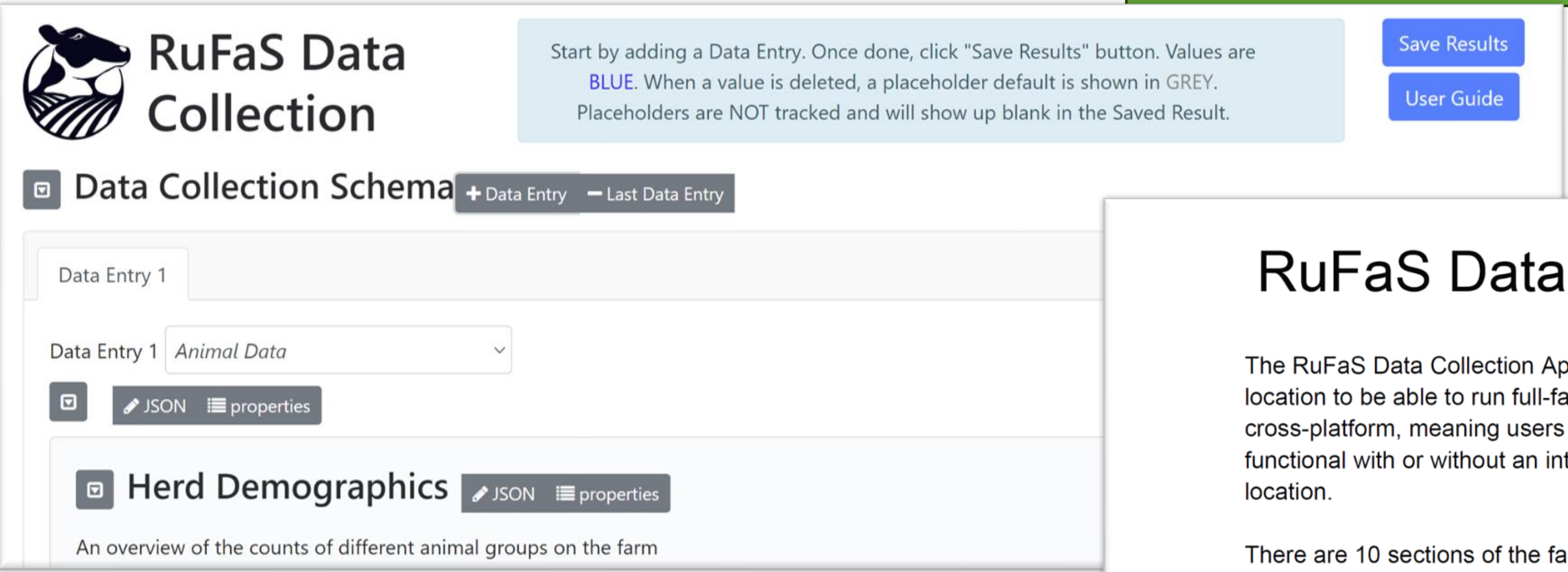
Returns all the data generated daily by different manure management components during the whole simulation.

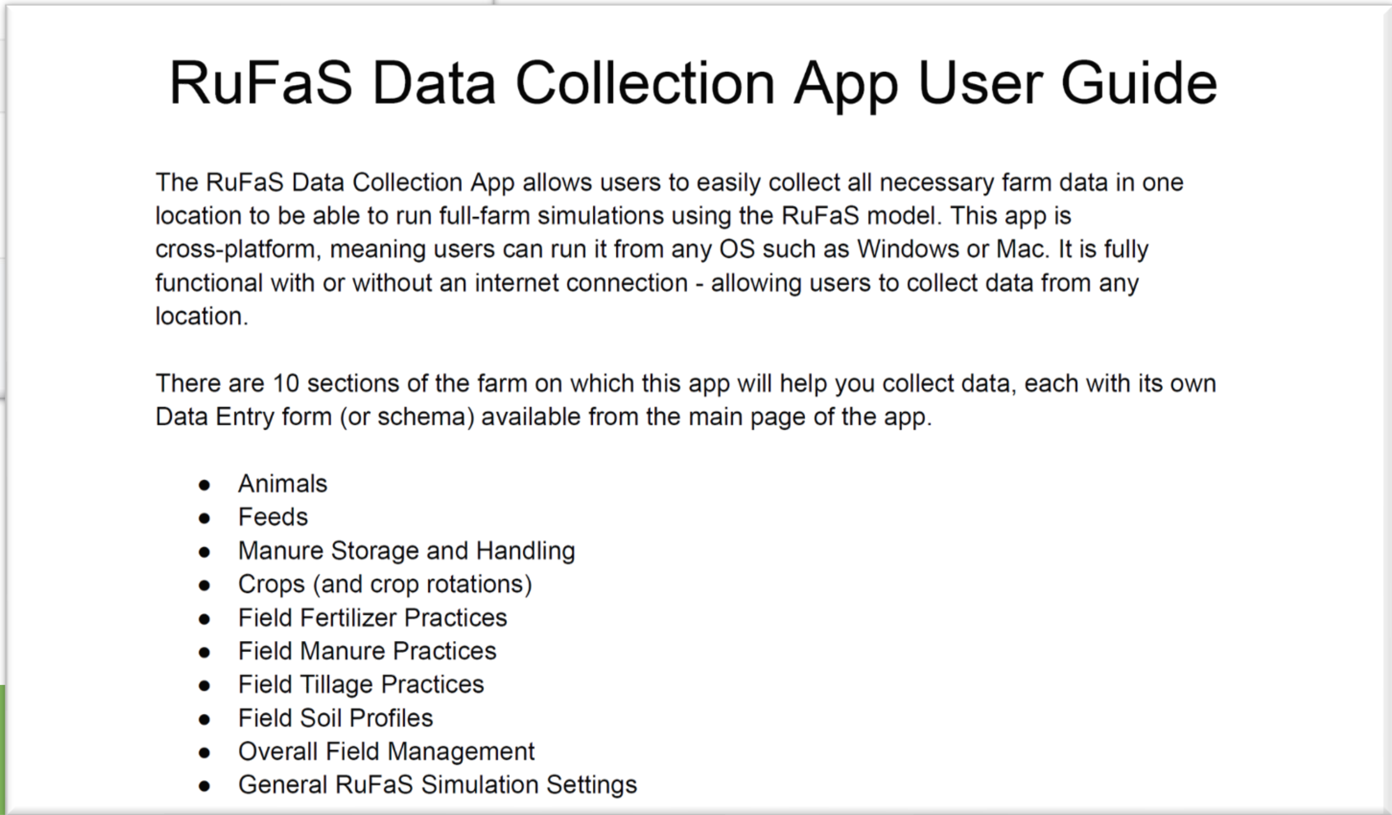
Returns:

A dictionary that stores all the data generated daily by the four main manure management components. Its structure is as follows:

USER INPUTS TO MODEL INPUTS

- Data collection app provides a more user friendly way to input data, including documentation





RuFaS Data Collection App User Guide

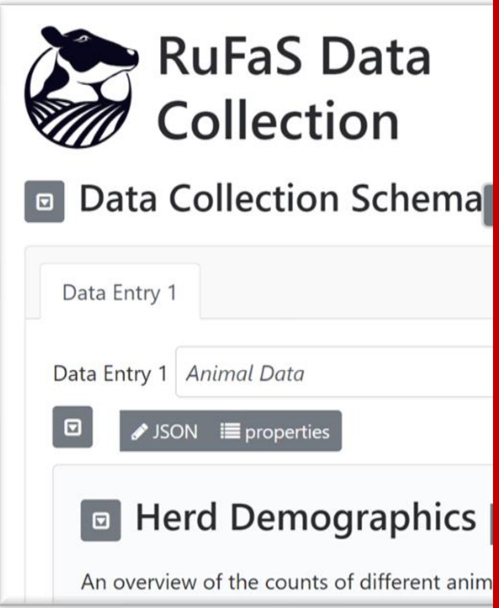
The RuFaS Data Collection App allows users to easily collect all necessary farm data in one location to be able to run full-farm simulations using the RuFaS model. This app is cross-platform, meaning users can run it from any OS such as Windows or Mac. It is fully functional with or without an internet connection - allowing users to collect data from any location.

There are 10 sections of the farm on which this app will help you collect data, each with its own Data Entry form (or schema) available from the main page of the app.

- Animals
- Feeds
- Manure Storage and Handling
- Crops (and crop rotations)
- Field Fertilizer Practices
- Field Manure Practices
- Field Tillage Practices
- Field Soil Profiles
- Overall Field Management
- General RuFaS Simulation Settings

USER INPUTS TO MODEL INPUTS

- Data collection app provides a more user friendly way to input data, including documentation



New Pro-Dairy Model Support Specialist working to Improve User Input Experience

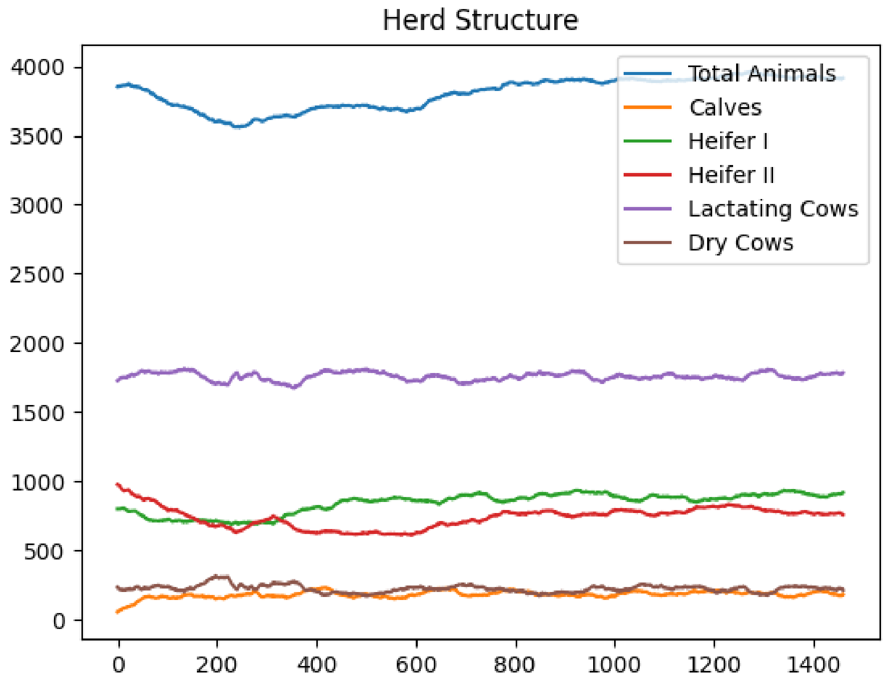
FUNCTIONALITY

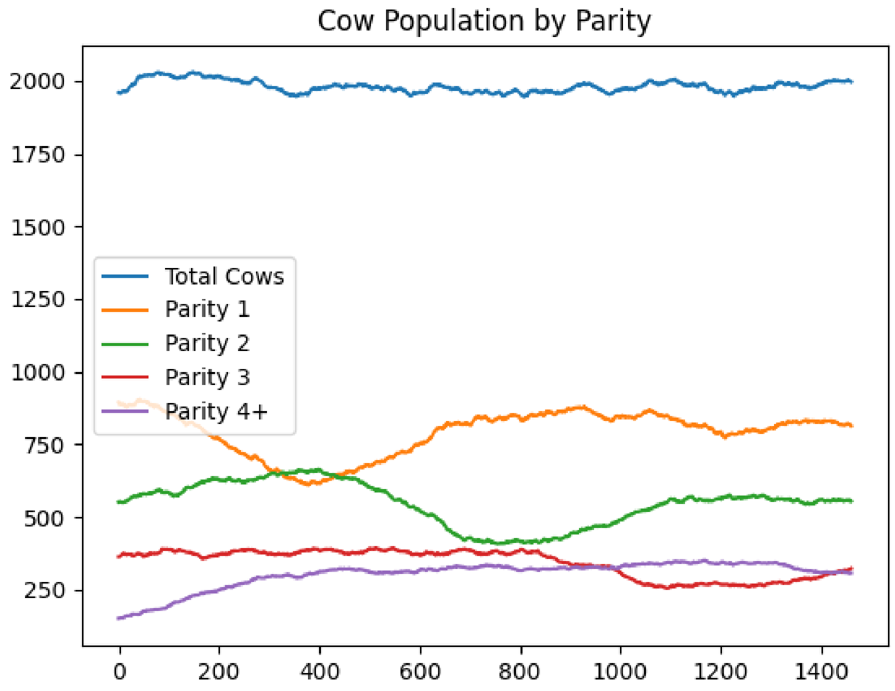
Animal Module

Management Options	Outcomes

✓ Tiestall, freestall, drylot, and compost-bedded pack barn housing
✓ Customized repro protocols for cows and heifers
✓ Diets with automated or user-defined ration formulation
✓ Flexible pen distribution and grouping
✓ Enteric methane mitigation supplements
    ✓ 3-NOP
    ❑ Monensin, EO, Seaweed
	✓ Milk and animal production
	✓ Feed use
	✓ Embedded Feed Emissions
	✓ Enteric methane
	✓ Manure production and composition
	❑ Energy Use
	❑ Water Use

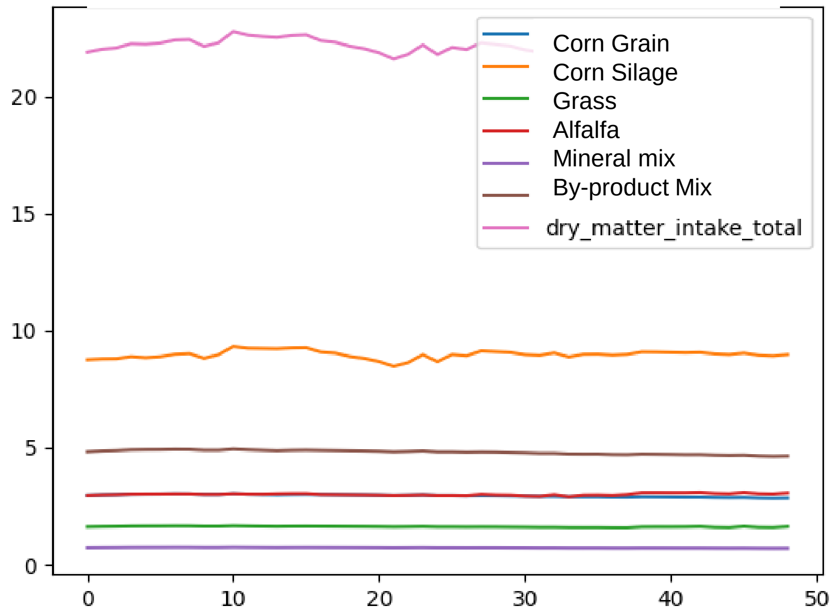
Herd Demographics Tracked Daily and Respond to Reproduction and Herd Exit Managment





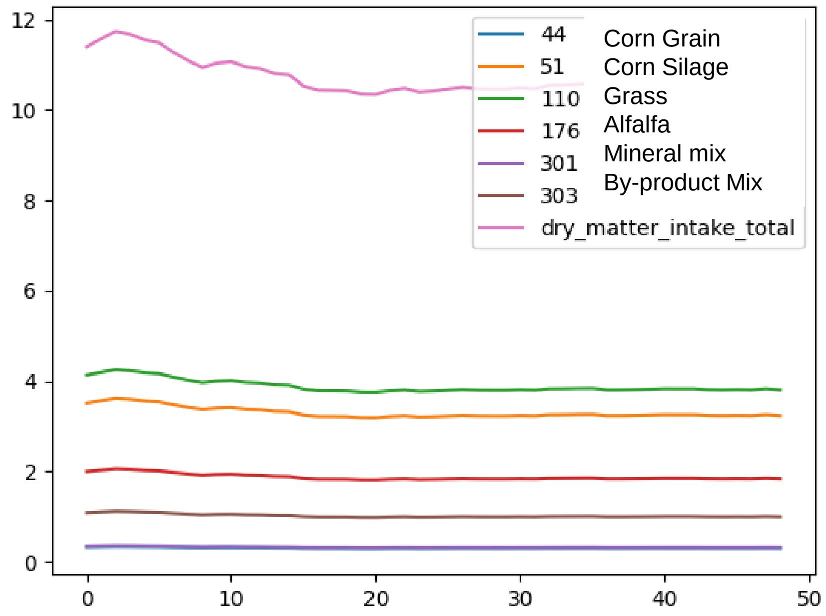
Intakes and Diets Assigned by Users or through Least Cost Formulation by Pen

Lactating Cow Diet and Intake



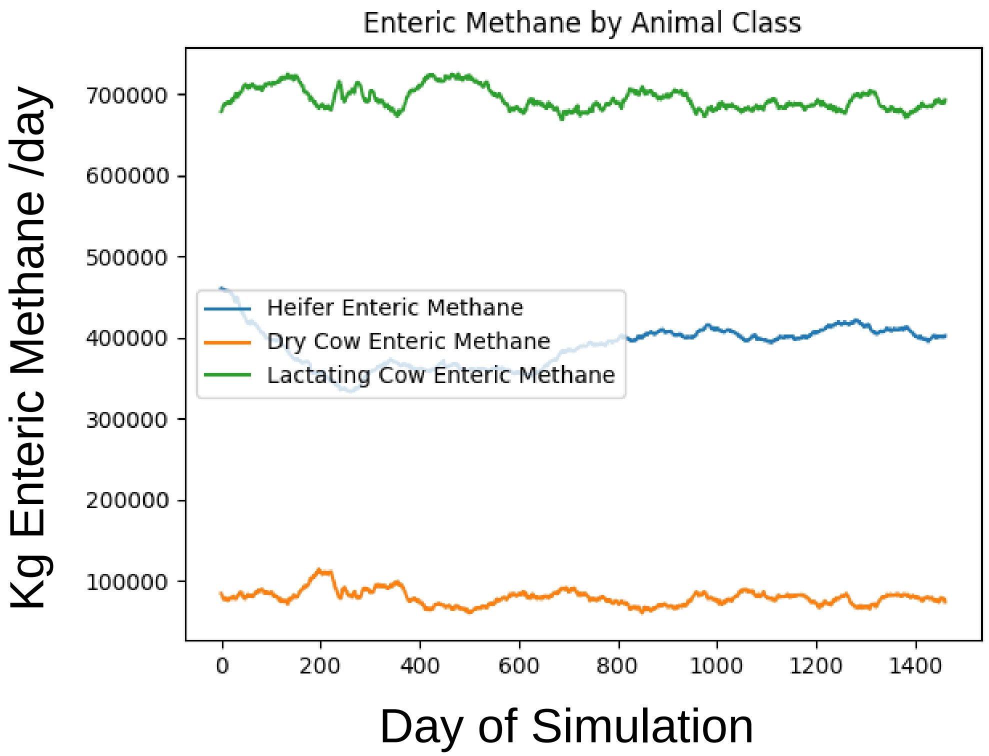
Month of Simulation

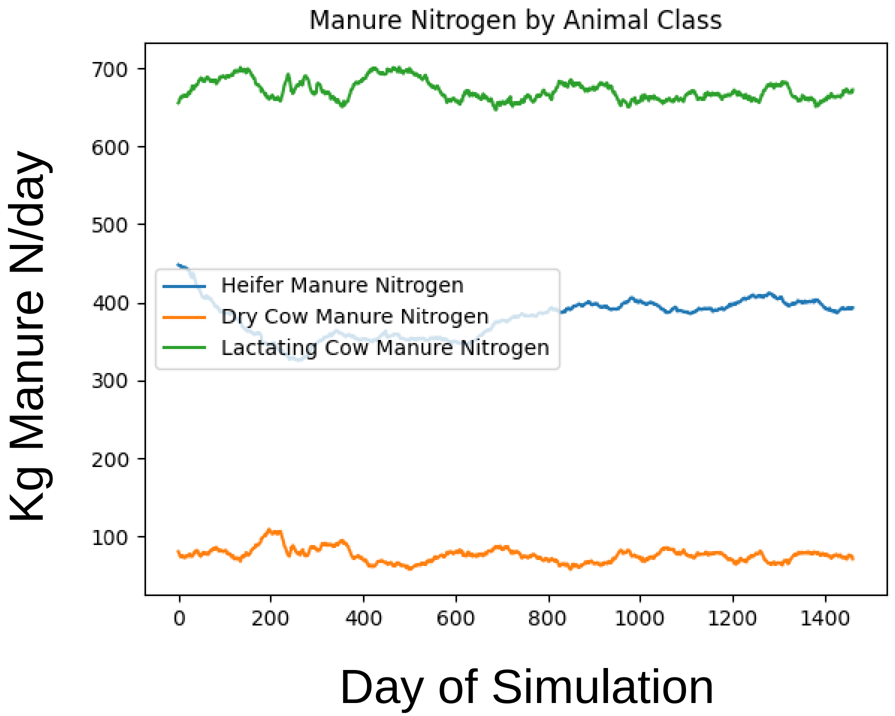
Heifer Pen Diet and Intake



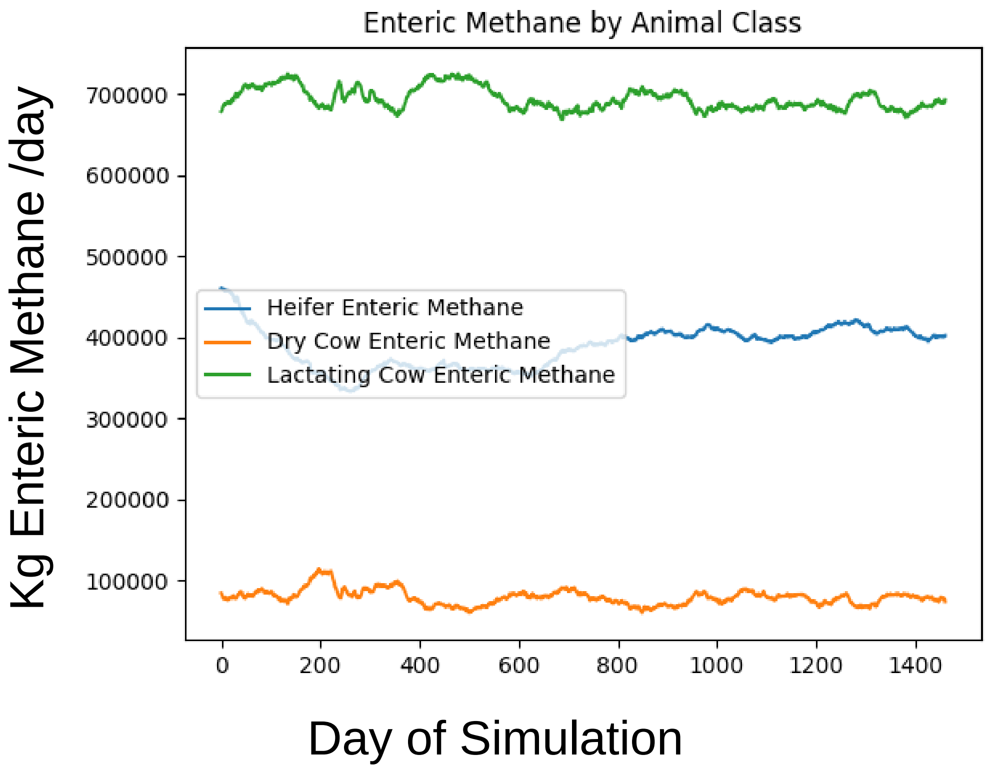
Month of Simulation

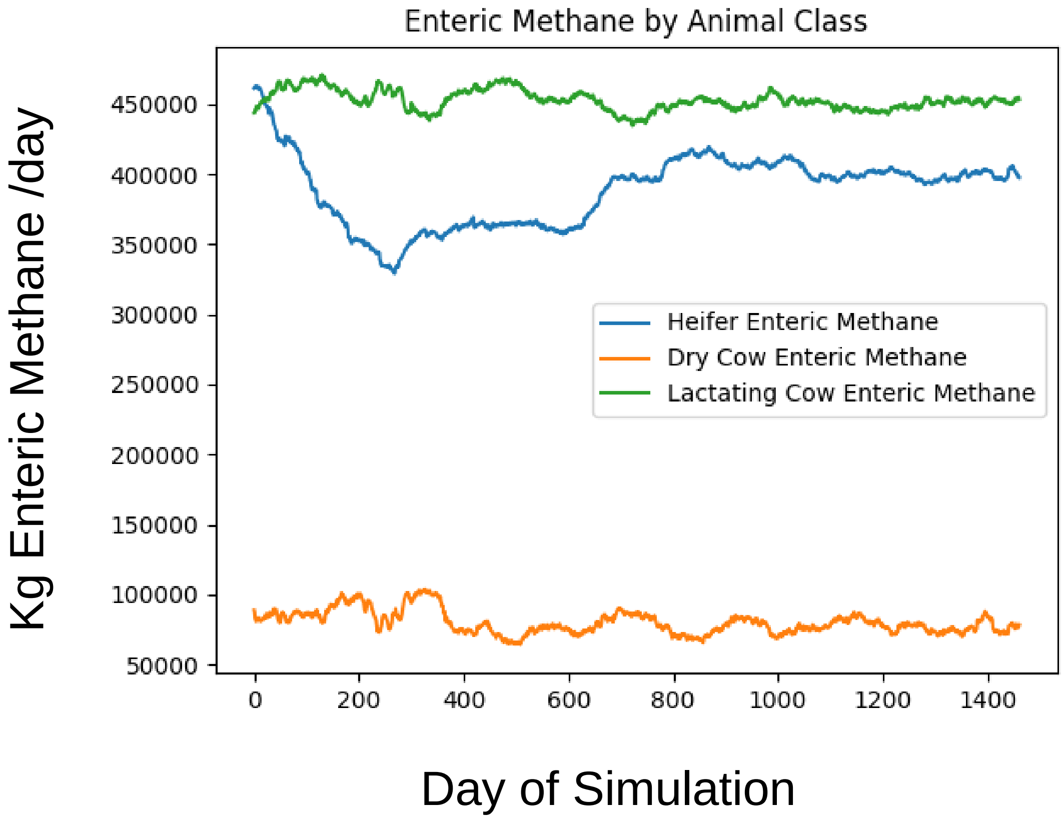
Enteric Methane and Manure Excretion Summed over All Animals by Class or Pen





Enteric Methane Mitigation for Lactating Cows Only: 3-NOP Example



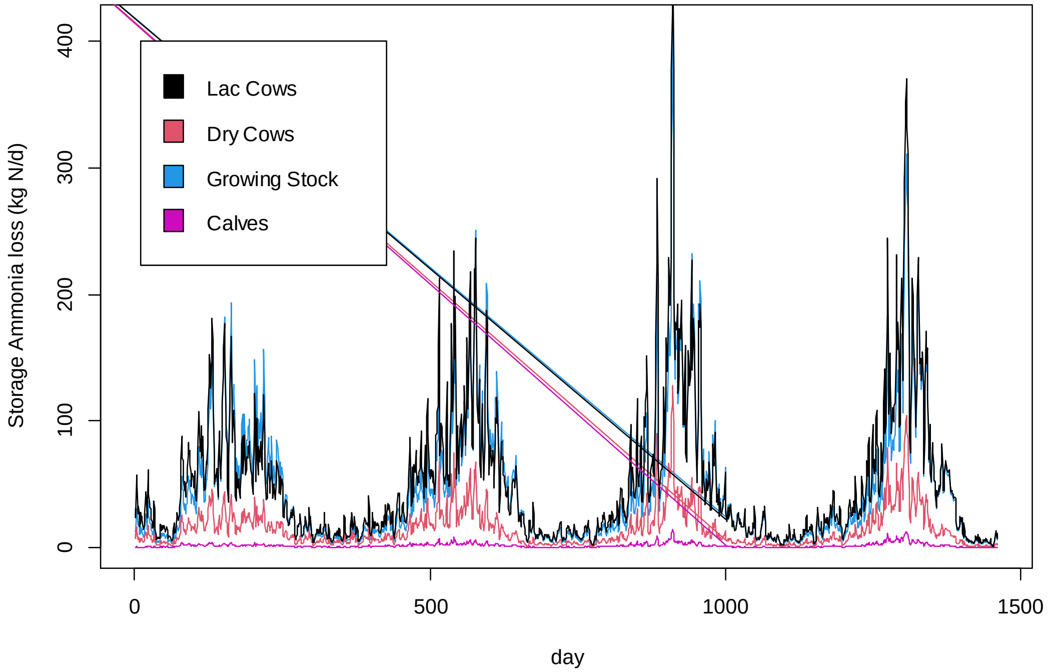


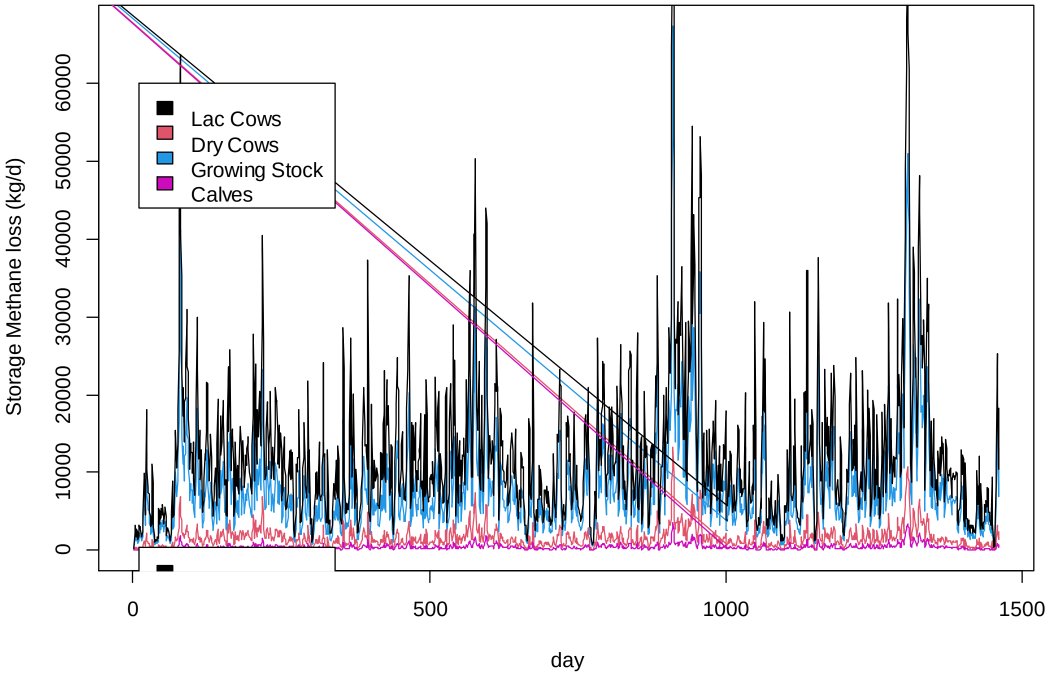
Manure Module

Management Options	Outcomes
✓ Collect manure from Animal module ✓ Transfer to Soil and Crop module ✓ Bedding types ✓ Scraping + Flushing ✓ Solid-Liquid Separation ✓ Anaerobic Digestion ✓ Long term storage liquid manure storage ✓ Compost-Bedded Pack barns ✓ Open lots ✓ Composting Storage	✓ N_2O, NH_3, and CH_4 emissions ✓ Manure composition tracked and updated throughout system ✓ Water use ❑ Energy use

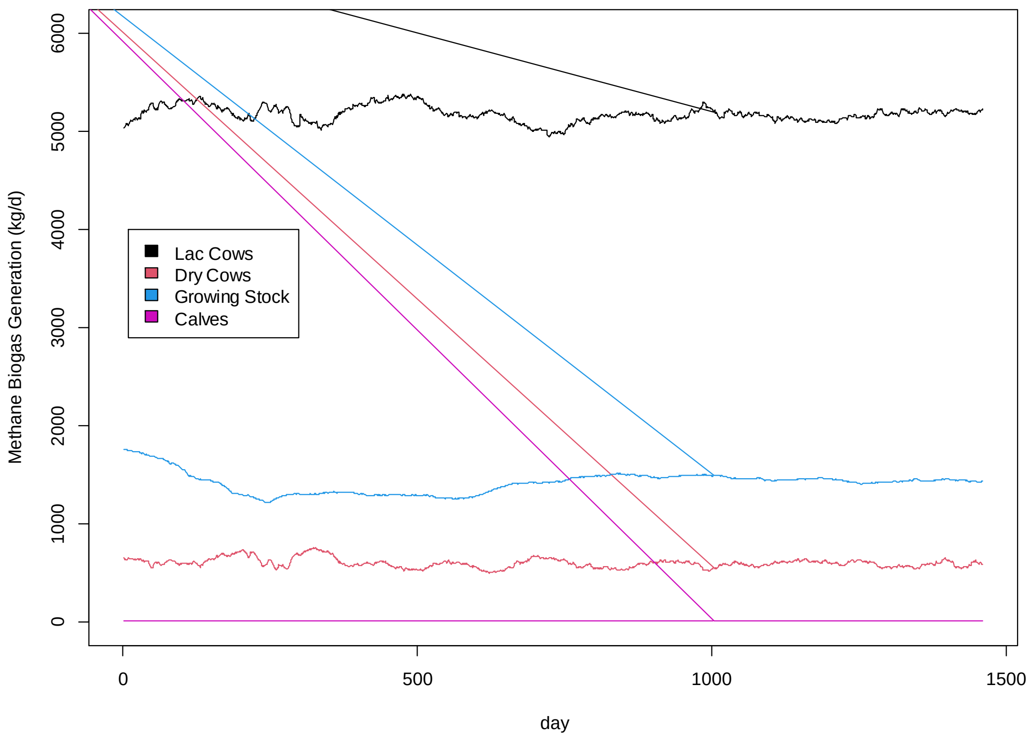
Storage Ammonia and Methane Losses

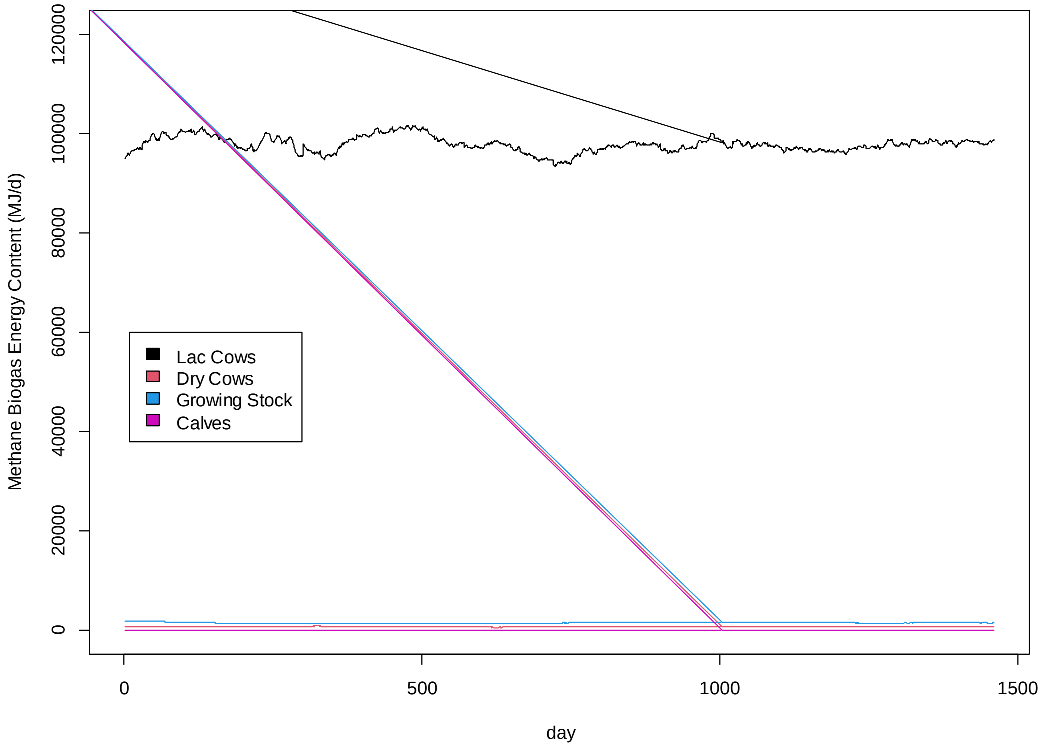
Outdoor Slurry Storage





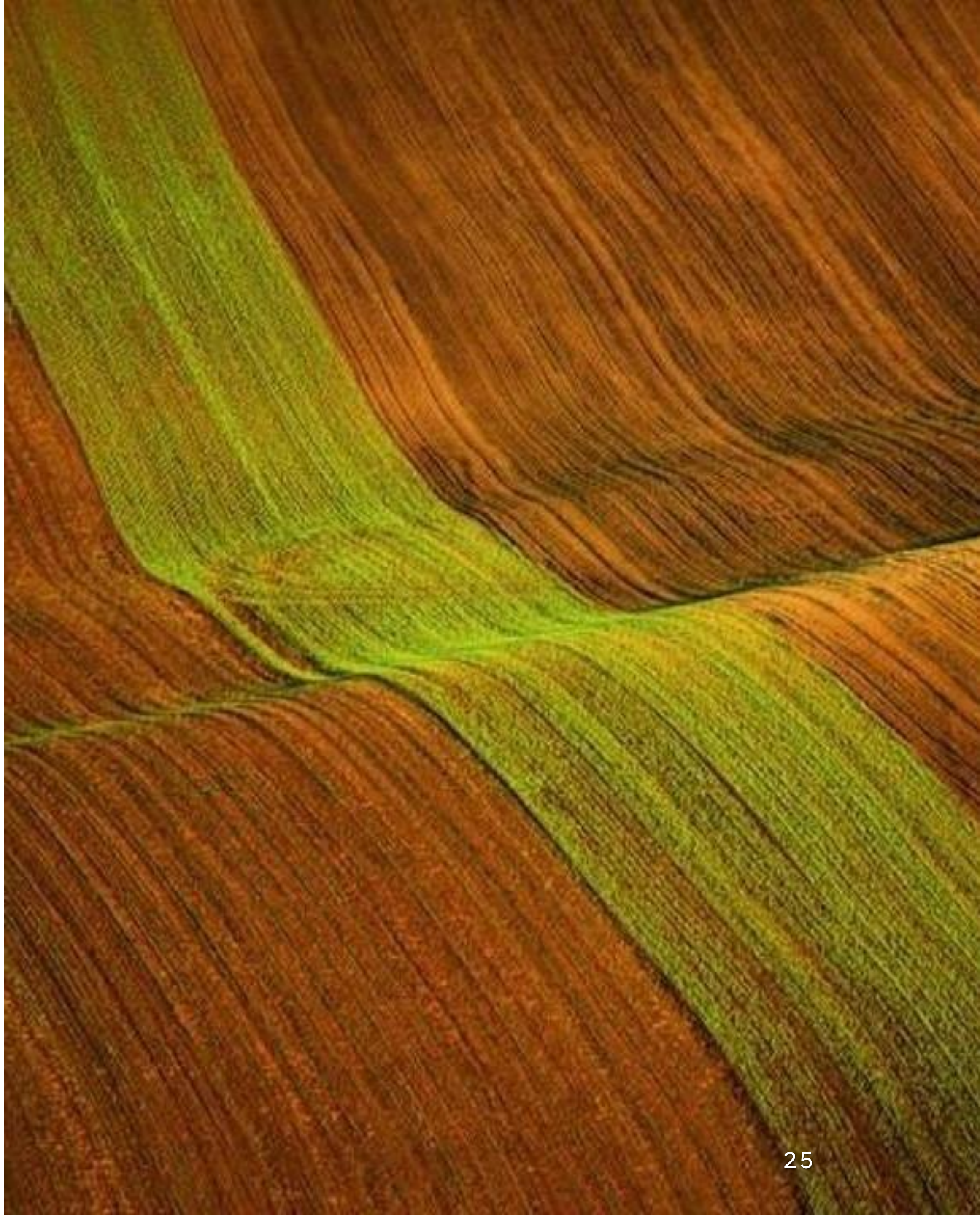
Anaerobic Digestion Biogas Generation





Soil and Crop Module

Management Options	Outcomes
✓ Variety of Dairy Crop types ✓ Cover cropping ✓ Range of Tillage practices ✓ Variable fertilizer and Manure application ✓ Irrigation	✓ N_2O, NH_3, and CO_2 Emissions ✓ N & P Leaching and Runoff ✓ Water use ✓ Crop yields and compositions ✓ Soil C dynamics ❑ Energy/Fossil Fuel Use



Feed Module

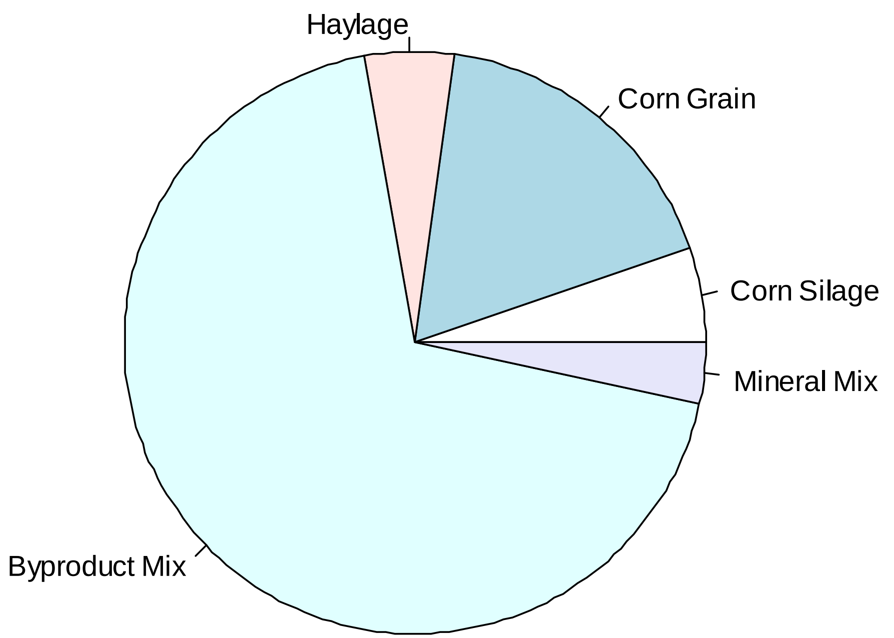
Management Options	Outcomes
✓ Silage, Hay, Baleage Storage ✓ Purchased feeds ✓ Inventory Tracking	✓ Embedded emissions in purchased feeds ❑ Energy/Fossil Fuel Use

Feed Emissions Estimates Higher than Previous works

Typical North East Diet:
1.52 kg CO2-eq/kg DM

Simulated Feed Emissions intensity:
0.7 -1.0 kg CO2-eq/ kg FPCM

Proportion of Feed Emissions Per Feed



IMMEDIATE GOALS

Evaluation & Sensitivity Analyses

- Across all modules and as a whole model
- Pilot Testing

Functional Requirements for V1

- Improvements in data synthesis and summaries
- Energy estimations

BEYOND V1

Add management practices

- Grazing
- Welfare
- Genetic Selection

Improve predictions as new data becomes available

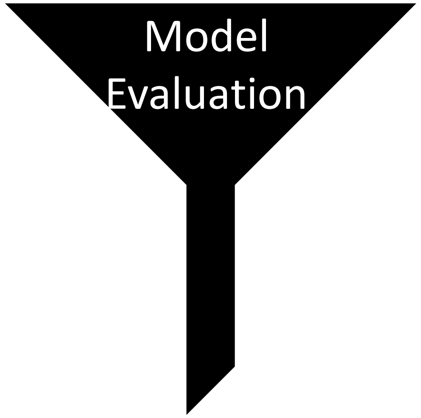
- Soil Carbon Model
- Manure N dynamics
- Enteric Methane Mitigation

User Interface/Accessibility

Pursuing two strategies to improve usability

Data Integration and interoperability

Filter Results Based on Influence



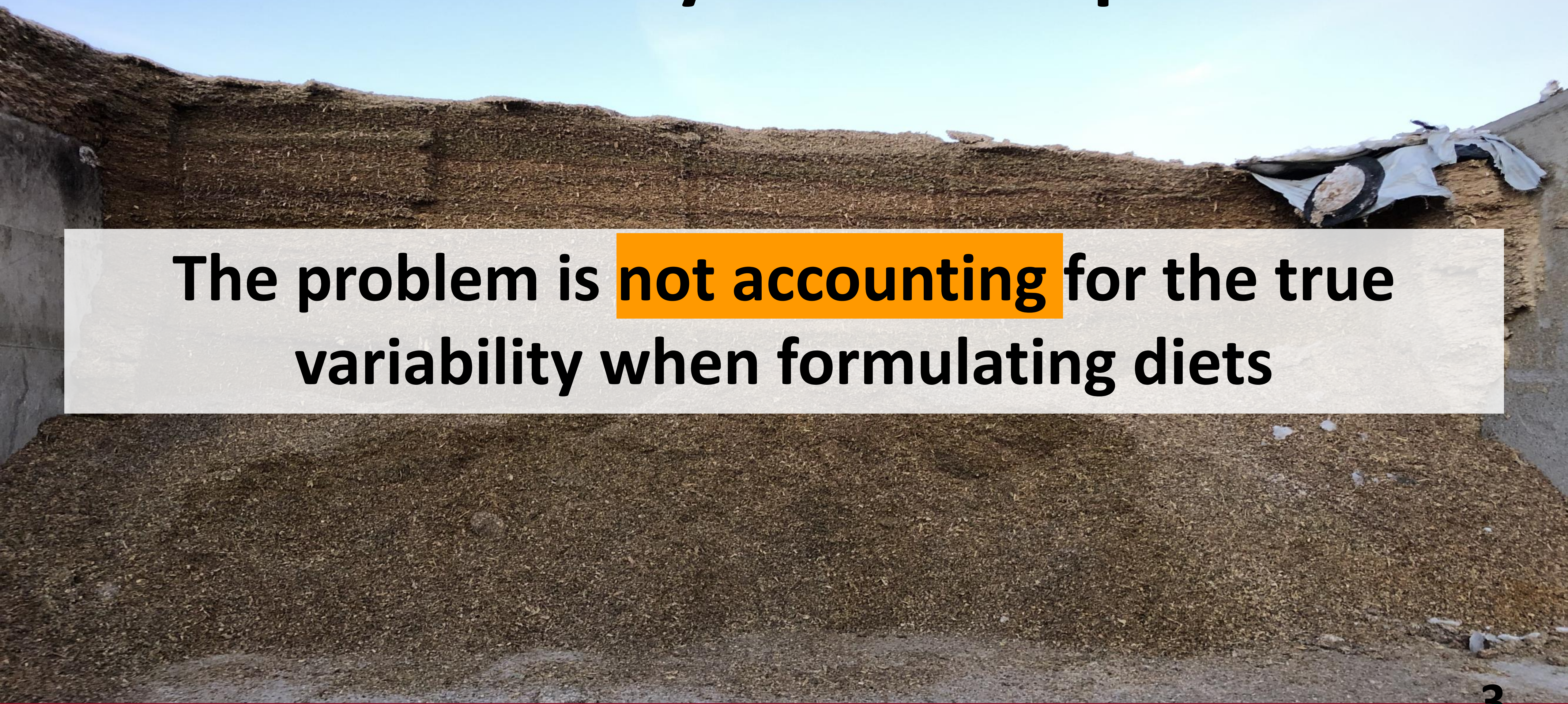
1. Essential inputs (30%)
2. Regional Default Values (20%)
3. Literature Based Default Values
4. Non-essential inputs/ constants

Managing Feed Variability

The only constant in life is change

~Heraclitus

“True variability is not the problem”



The problem is not accounting for the true variability when formulating diets

Practices to manage diet variability

1. **Over-formulating** CP, NEL, ME, MP

2. **Proactive:**
   - Sampling more frequently
   - Reformulate diets more frequently.

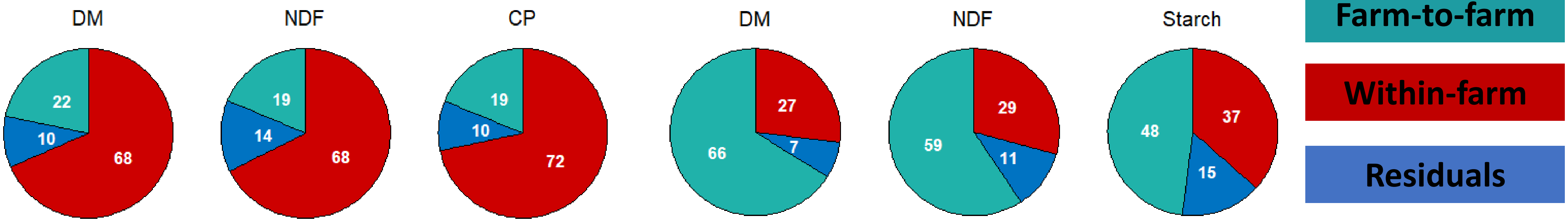
1. **Reactive:**
   - Decrease in milk yield
   - Change in MUN

Objectives

Quantifying variability → **Optimizing** sampling practices → **Monitoring** forage composition variability → **Evaluate**

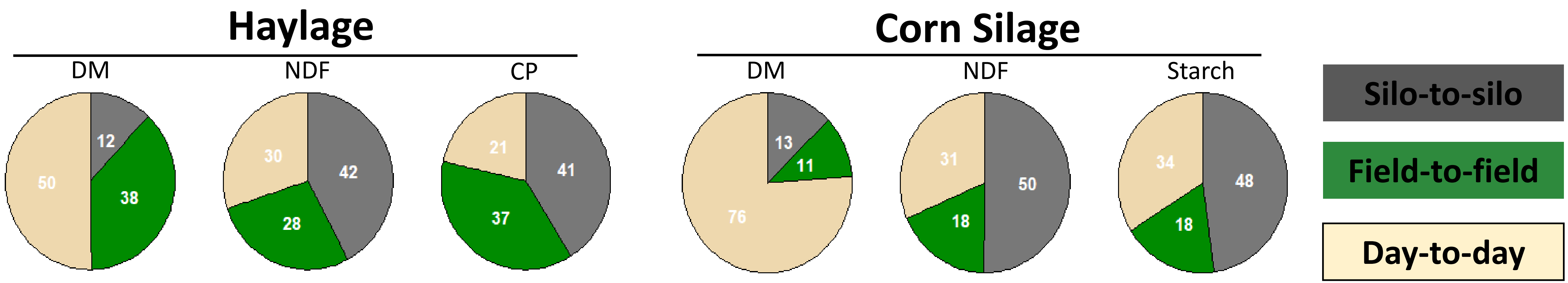
Partitioning of total variation by source at feed-out

Haylage | Corn Silage



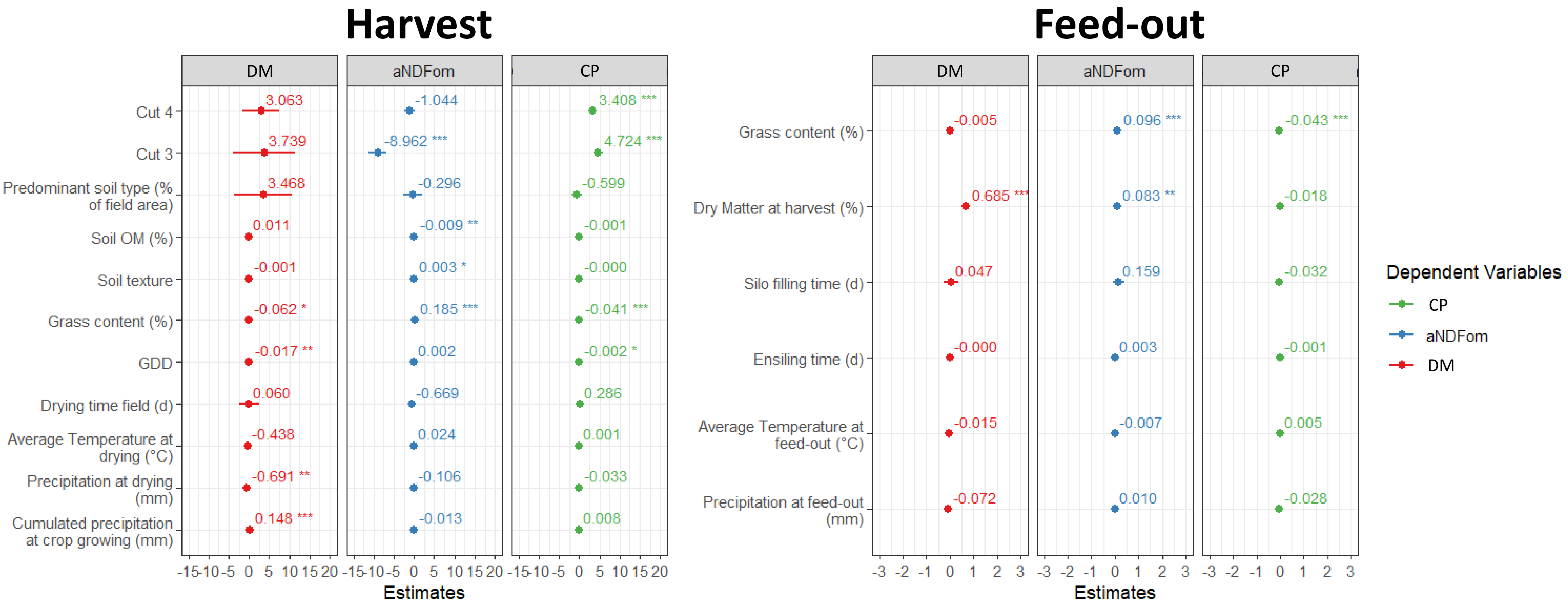
	Haylage DM	Haylage NDF	Haylage CP	Corn Silage DM	Corn Silage NDF	Corn Silage Starch
σ^2	51.58	17.22	5.81	18.17	22.56	25.22

Partitioning within-farm variation by source at feed-out

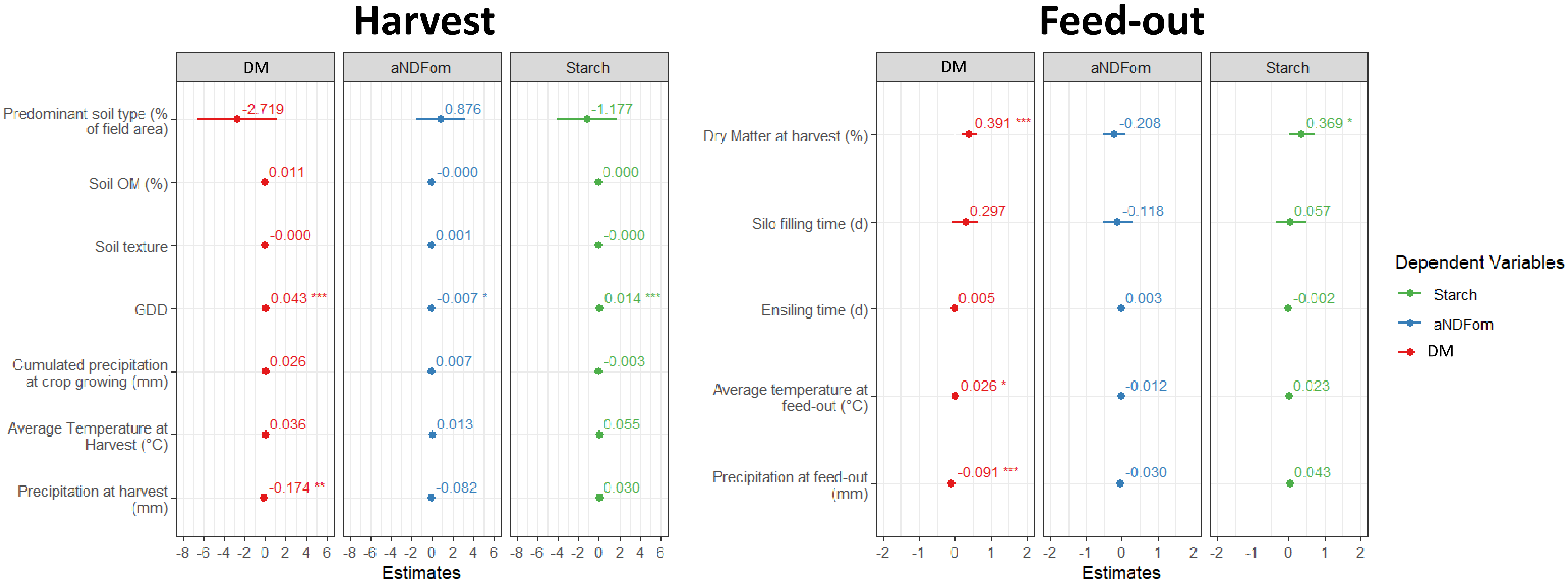


σ^2	35.32	11.65	4.16	4.91	6.60	9.24

Fixed effects for production of haylage



Fixed effects for production of corn silage



Take home message

1. Silo, Day, Field are important sources of variability

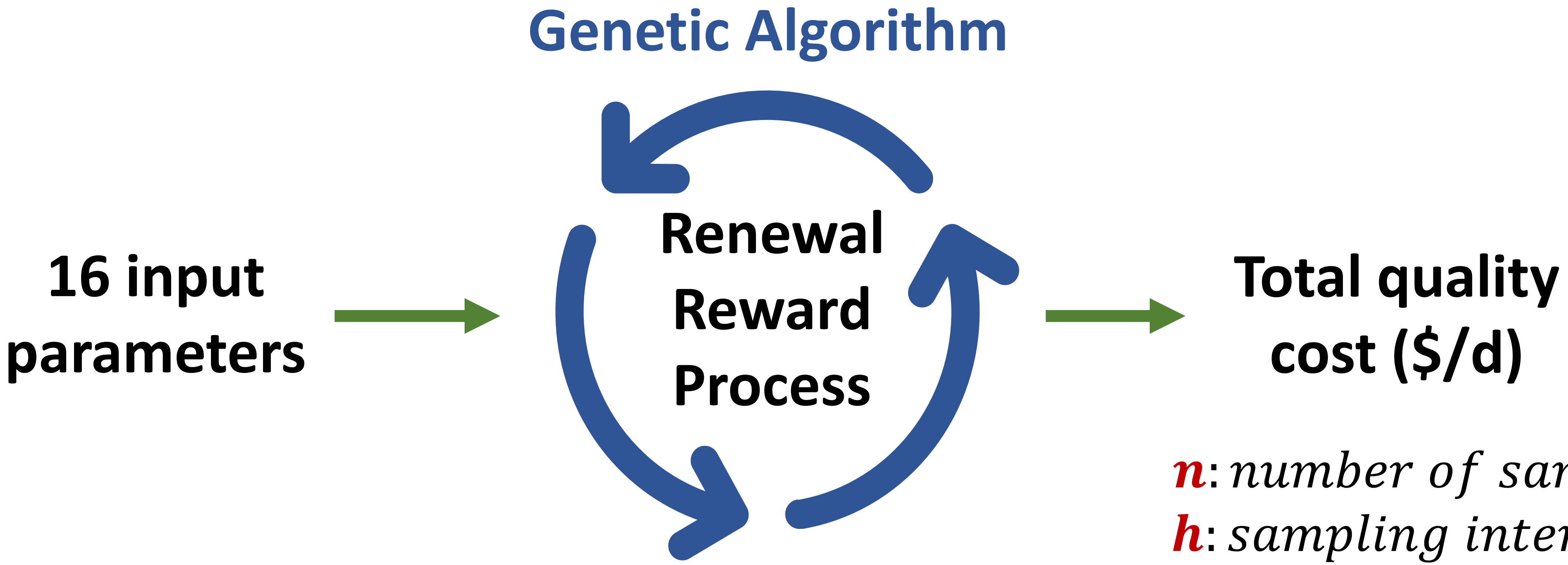
2. Collect samples from individual silos

3. Collect 2 or more independent samples

4. Optimize the sampling protocols within-silo

Optimizing sampling protocols

Optimizing sampling protocols



$\boldsymbol{n}$: *number of samples*
$\boldsymbol{h}$: *sampling interval*
$\boldsymbol{L}$: *control limits*

(St-Pierre and Cobanov, 2007)

Influential inputs

Genetic Algorithm

Herd size
Milk price
Time to sample and analysis
Cost per sample

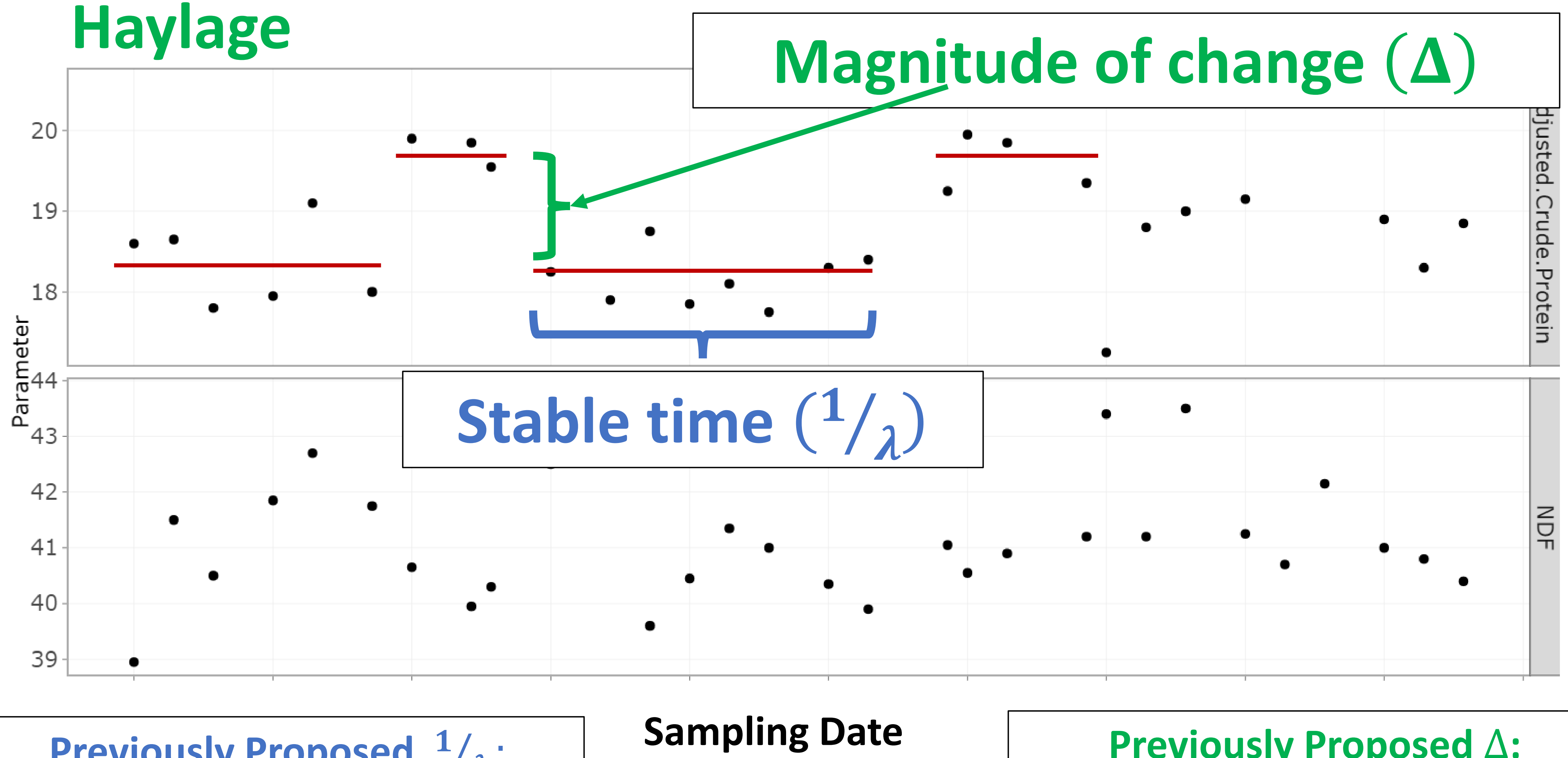
Stable time $({}^{1}/_{\lambda})$
Magnitude of change (Δ)
Change in milk yield (kg/d)

Total quality cost ($/d)

$\boldsymbol{n}: number\ of\ samples$
$\boldsymbol{h}: sampling\ interval$
$\boldsymbol{L}: control\ limits$

(St-Pierre and Cobanov, 2007)

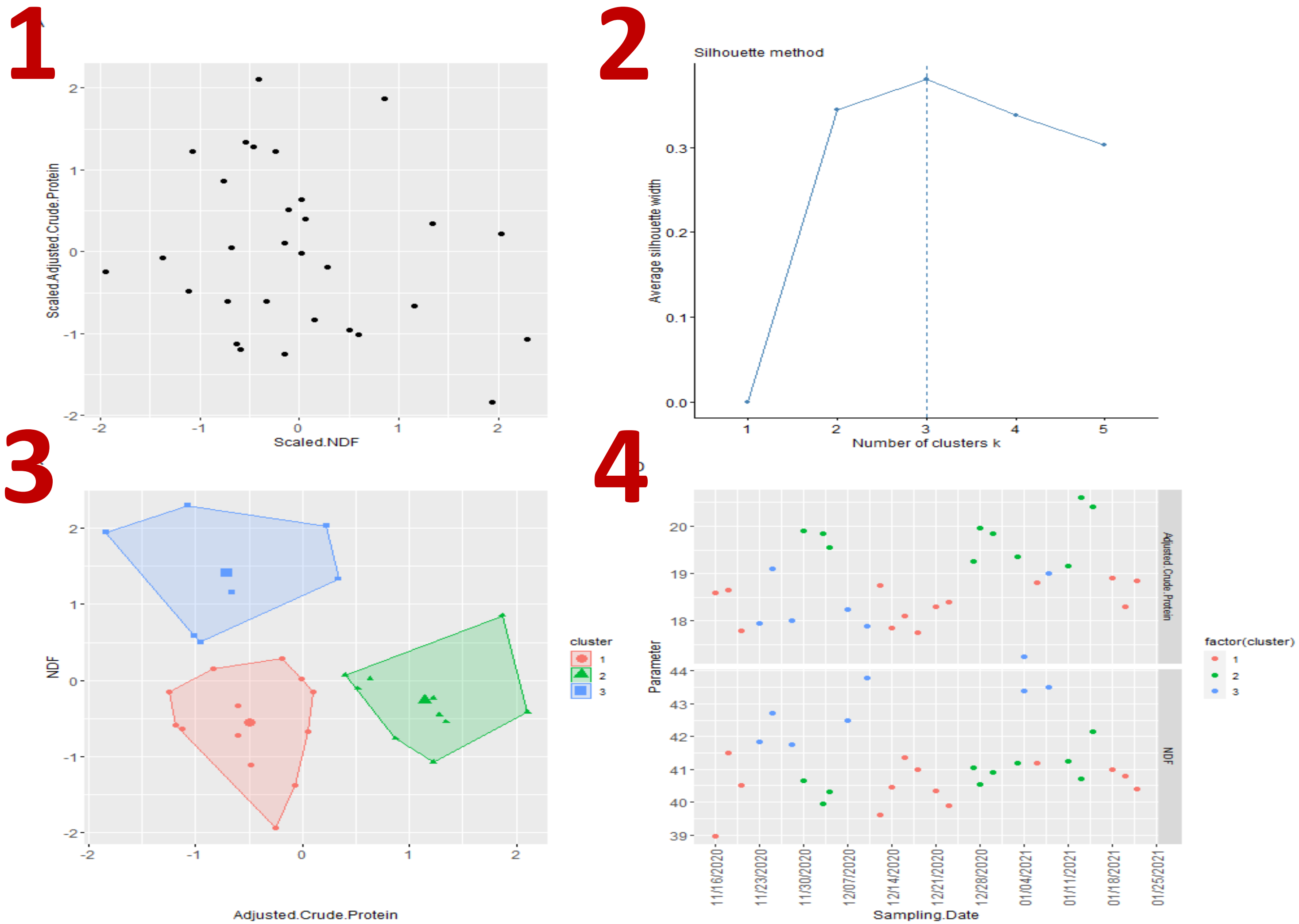
Haylage



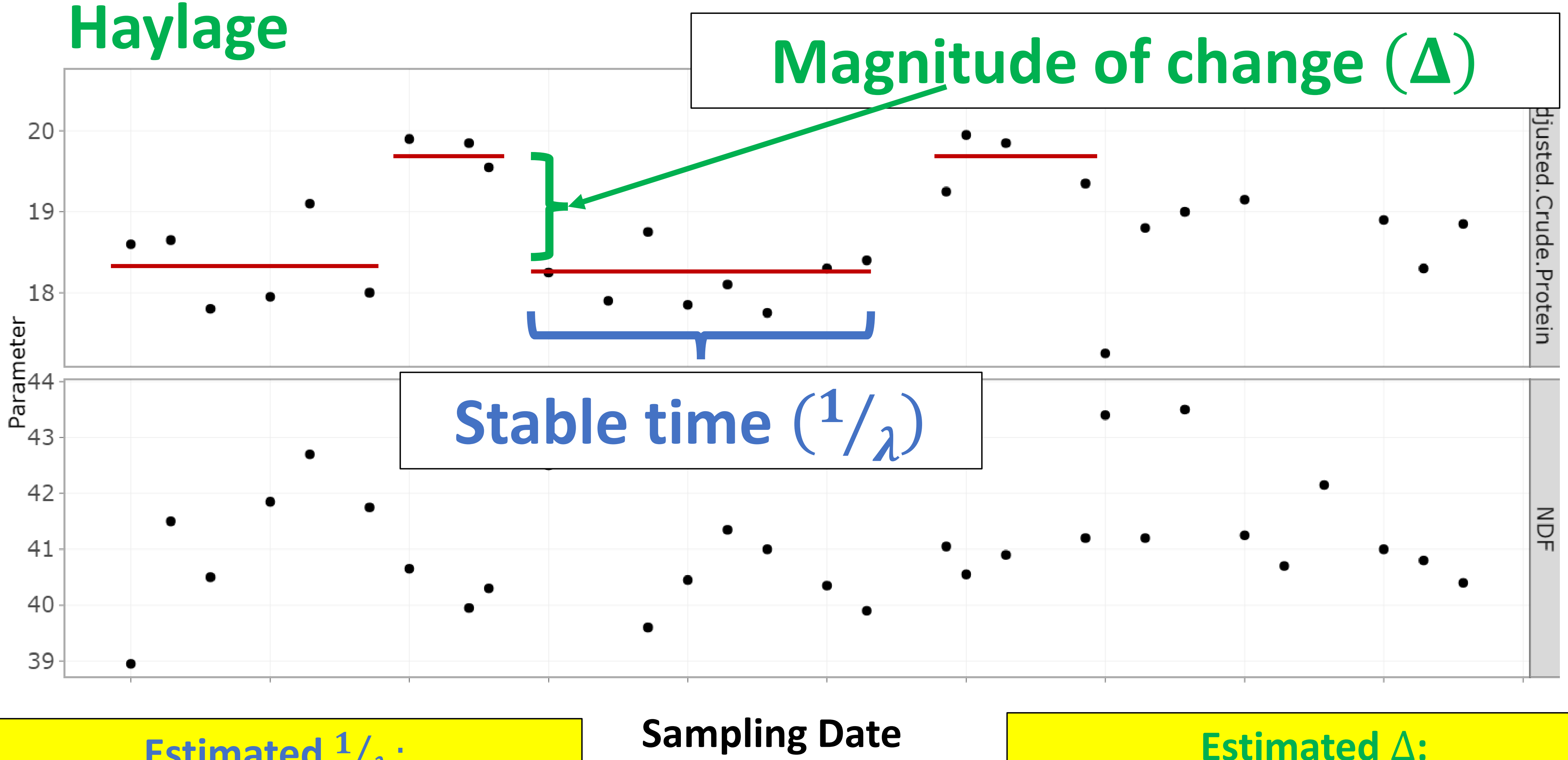
Previously Proposed $^1/_\lambda$:
30 Days

Previously Proposed Δ:
1.5 x SD

K-means clustering to estimate Δ and $1/\lambda$



Haylage



Estimated $^1/_\lambda$:
2 to 30 Days

Estimated Δ:
1.5 to 6.5 x SD

Optimal sampling practices

		Corn Silage			Haylage		
Optimal sampling scenario	**Farm size**	**h**	**n**	**L**	**h**	**n**	**L**
Default $(\widehat{1/\lambda} = 30\text{ d}, \widehat{\Delta} = 1.5)$	100	12	2	1.13	12	2	1.13
	300	-	-	-	6	2	1.24
	500	5	2	1.18	5	2	1.18
	600	5	2	1.15	5	2	1.15
	700	4	2	1.25	4	2	1.25
	1000	3	2	1.23	3	2	1.23
	2000	2	2	1.33	2	2	1.33
	3000	2	3	1.42	2	3	1.42
K-means cluster	100	10	2	4.70	10	2	3.59
	300	-	-	-	5	2	2.57
	500	4	2	1.66	5	2	0.35
	600	3	2	2.66	4	2	0.86
	700	3	2	1.78	4	2	2.64
	1000	2	2	1.96	3	2	3.99
	2000	2	2	1.65	2	2	2.25
	3000	2	2	2.73	2	2	2.43

Evaluation

Treatment structure

Treatment protocol

Optimal sampling and monitoring protocol

Parameter	Value
Herd size	2000
Forage	Haylage and Corn silage
Milk price $/kg	$0.34
Cost of Lab analysis ($/lab)	$25
$^{1}/_{\lambda}$ (d)	4
Δ (SDs)	1.5
Number of samples	**2**
Sampling interval (d)	**2**
Factor to estimate the limits of variation	**0.831**

Control protocol

Sampling and diet formulation practices of the farm

Collected 2 independent samples from haylage, corn silage, and TMR 3x per week for 16 weeks

Summary of the quality control analysis

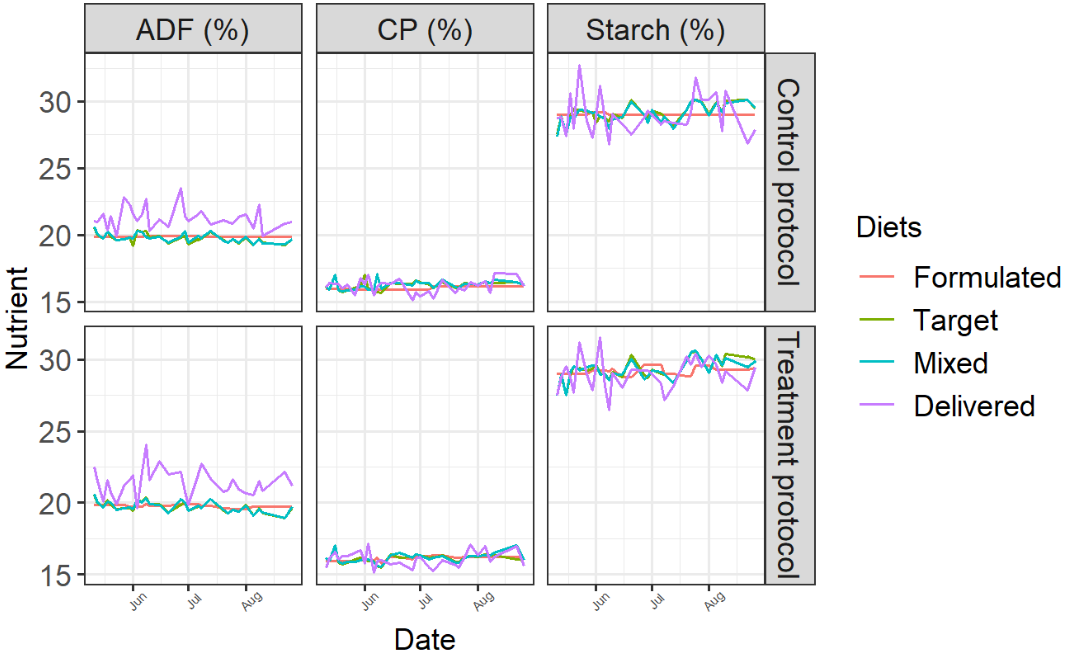
Forage	Monitored Nutrient	Deviation between stable groups (SD)	Reported changes in components	Total False alarms
Haylage	CP	1.16	11	13
Corn Silage	Starch	1.94	12	7

Formulated diets

	Treatment protocol	Control protocol
Reformulation (n)	**13**	**5**
Reformulation interval (d)	**8**	**22**
Ingredient (kg of DM)		
Haylage	5.51 ± 0.09	5.48 ± 0.09
Canola	1.06 ± 0.06	1.16 ± 0.08
Corn Meal	4.86 ± 0.13	4.82 ± 0.07
Corn Silage	9.21 ± 2.51	9.20 ± 2.5
Premix	3.55 ± 0.25	3.56 ± 0.25
Soybean Meal	1.30 ± 0.00	1.30 ± 0.00
Whey	0.40 ± 0.02	0.40 ± 0.02

Impacts on Diet Accuracy

Diets	Component	Control protocol	Treatment protocol
|Formulated – Target|	CP (%)	0.245	0.135
	ADF (%)	0.274	0.246
	Starch (%)	0.396	0.55
|Target – Mixed|	CP (%)	0.048	0.036
	ADF (%)	0.054	0.045
	Starch (%)	0.063	0.071
|Mixed – Delivered|	CP (%)	0.44	0.415
	ADF (%)	1.324	1.578
	Starch (%)	0.836	0.768



Mixed model analysis outputs

Parameter	Control protocol	Treatment protocol	SE	P-value
Milk yield (kg/cow/d)	45.10	46.18	0.46	**0.099**
DMI (kg/cow/d)	25.16	25.57	0.27	0.229
Diet Forage (%)	58.89	58.94	0.06	0.439
FE (kg Milk yield/kg DMI)	0.82	0.83	0.02	0.659
Diet cost ($/cow/d)	$7.40	$7.66	0.06	**0.025**
IOFC ($/cow/d)	$16.13	$16.33	0.32	0.583

Take home message

1. Monitoring forages increased the reformulation frequency.

2. The treatment protocol improved the accuracy of the CP and ADF content of the target diet and mixed diet.

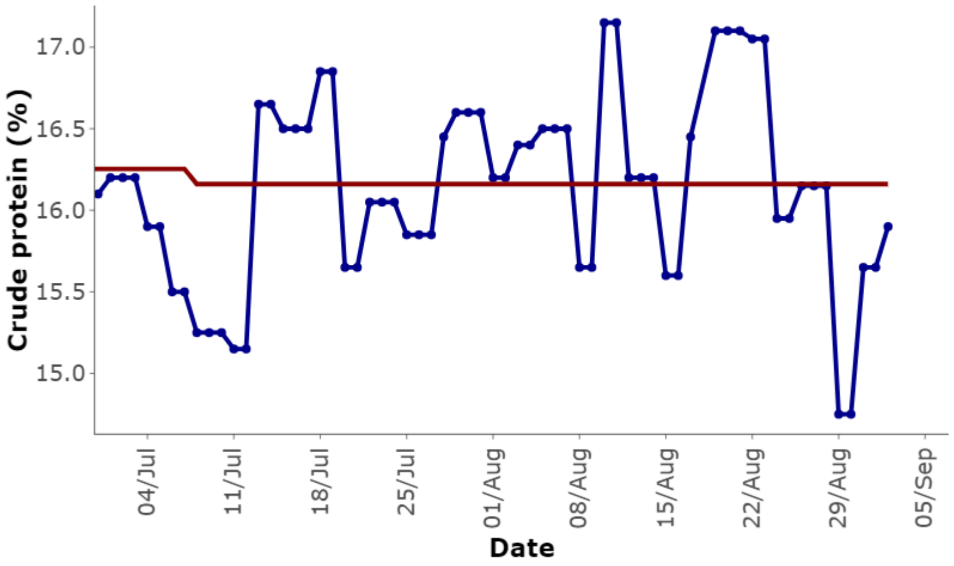
3. The increased accuracy in CP is a likely cause of the increased tendency of milk yield.

Future work

1. Expand model and algorithms to include all feeds and relevant nutrients.

2. Work with industry partners to increase the number of farms and study interval for future on farm-evaluations

3. Integrate with current farm diet formulation and mixing software systems



Many thanks to all!

- NEAFA for their support and guidance
- Agriculture and Food Research Initiative competitive award no. 2020-68014-31466 from the USDA National Institute of Food and Agriculture
- Dairy Management, Inc
- The Cornell Atkinson Center for Sustainability
- General Mills
- Smith-Lever Award no. 2021-22-123
- USDA-DFRC

RuFaS.org
rufascornell@gmail.com
kfr3@cornell.edu