



เอกสารประกอบการฝึกอบรมเชิงปฏิบัติการ เรื่อง

“การใช้งาน Cloud-Based IrrisAT Application และการสร้าง
แบบจำลองสำหรับงานบริหารจัดการเขื่อนด้วยโปรแกรม MATLAB” &
“การบริหารจัดการเขื่อนด้วยเทคนิคปัญญาประดิษฐ์เบื้องต้นด้วย
ภาษา R และ MiniZinc”

วันที่ 12-13 พฤศจิกายน พ.ศ. 2563 เวลา 09.00-16.30 น.
ห้องปฏิบัติการคอมพิวเตอร์ ชั้น 1 สำนักงานอธิการบดี มหาวิทยาลัยมหิดล

รศ.ดร.อาริยา ฤทธิมา

ภาควิชาวิศวกรรมโยธาและสิ่งแวดล้อม คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล

อ.ดร.ยุภนา พันธุ์มงคลศิลป์

สาขาวิชาวิศวกรรมสิ่งแวดล้อมและการจัดการภัยพิบัติ มหาวิทยาลัยมหิดล วิทยาเขตกาญจนบุรี

อ.ดร.อรรณย์ ศรีรัตนากาญจนอน

คณะสิ่งแวดล้อมและทรัพยากรศาสตร์ มหาวิทยาลัยมหิดล

อ.ดร.วุฒิชชาติ แสงผล

คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล

อ.ดร.จิตาภา ไกรสังข์

คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล

อ.ดร.ยุภนา ตาละลักษมณ์

ภาควิชาวิศวกรรมทรัพยากรน้ำ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

คณะนักวิจัย



กำหนดการฝึกอบรมเชิงปฏิบัติการ

“การใช้งาน Cloud-Based IrrisAT Application และการสร้างแบบจำลองสำหรับงานบริหาร
จัดการเขื่อนด้วยโปรแกรม MATLAB”

ห้องปฏิบัติการคอมพิวเตอร์ ชั้น 1 สำนักงานอธิการบดี มหาวิทยาลัยมหิดล

วันที่ 12 พฤศจิกายน พ.ศ. 2563

เวลา	กำหนดการ
09.00–09.30 น.	ลงทะเบียน
09.30–09.40 น.	กล่าวต้อนรับ : ผู้ช่วยศาสตราจารย์ ดร.ณวัชร สุรินทร์กุล หัวหน้าภาควิชาวิศวกรรมโยธาและสิ่งแวดล้อม คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล กล่าววัตถุประสงค์การฝึกอบรมเชิงปฏิบัติการ : รองศาสตราจารย์ ดร.สุจิตต์ คุณธนกุลวงศ์ ประธานแผนงานยุทธศาสตร์เป้าหมาย โครงการวิจัยเข้มมุ่ง สำนักประสานงานวิจัยการจัดการน้ำเชิงยุทธศาสตร์ สกสว.
09.40–10.15 น.	นำเสนอแนวคิดและหลักการของโครงการวิจัยเรื่อง “โครงการกลยุทธ์การ ปรับเปลี่ยนแนวทางการปฏิบัติการอ่างเก็บน้ำสำหรับพัฒนาการบริหาร จัดการน้ำต้นทุนในระยะยาวของเขื่อนภูมิพล (ระยะที่ 1)” หัวหน้าโครงการวิจัย : รองศาสตราจารย์ ดร.อารีญา ฤทธิมา ภาควิชาวิศวกรรมโยธาและสิ่งแวดล้อม คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล
10.30–12.00 น.	สาธิตและฝึกปฏิบัติการการใช้งาน Cloud-Based IrrisAT Application สำหรับประเมินปริมาณความต้องการน้ำเพื่อการเกษตร วิทยากร : อาจารย์ ดร.ยุทธนา พันธุ์กมลศิลป์ สาขาวิชาวิศวกรรมสิ่งแวดล้อมและการจัดการภัยพิบัติ มหาวิทยาลัยมหิดล วิทยาเขตกาญจนบุรี
12.00–13.00 น.	รับประทานอาหารกลางวัน
13.00–16.00 น.	สาธิตและฝึกปฏิบัติการการสร้างแบบจำลองสำหรับงานบริหารจัดการ เขื่อนด้วยโปรแกรม MATLAB วิทยากร : อาจารย์ ดร.ยุทธนา พันธุ์กมลศิลป์ สาขาวิชาวิศวกรรมสิ่งแวดล้อมและการจัดการภัยพิบัติ มหาวิทยาลัยมหิดล วิทยาเขตกาญจนบุรี
16.00–16.30 น.	แลกเปลี่ยนและตอบข้อซักถาม

** เวลา 10.15–10.30 น. และ 14.15–14.30 น. รับประทานอาหารว่างและเครื่องดื่ม



กำหนดการฝึกอบรมเชิงปฏิบัติการ

“การบริหารจัดการเขื่อนด้วยเทคนิคปัญญาประดิษฐ์เบื้องต้นด้วยภาษา R และ MiniZinc”

ห้องปฏิบัติการคอมพิวเตอร์ ชั้น 1 สำนักงานอธิการบดี มหาวิทยาลัยมหิดล

วันที่ 13 พฤศจิกายน พ.ศ. 2563

เวลา	กำหนดการ
09.00–09.30 น.	ลงทะเบียน
09.30–12.00 น.	สาธิตและฝึกปฏิบัติการสร้างแบบจำลองการพยากรณ์น้ำด้วยภาษา R วิทยากร : อาจารย์ ดร.จิตาภา ไกรสังข์ คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล
12.00–13.00 น.	รับประทานอาหารกลางวัน
13.00–16.00 น.	สาธิตและฝึกปฏิบัติการการสร้างแบบจำลองสำหรับงานบริหารจัดการ เขื่อนด้วยโปรแกรม MiniZinc วิทยากร : อาจารย์ ดร.วุฒิชชาติ แสงวงผล คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล
16.00–16.30 น.	แลกเปลี่ยนและตอบข้อซักถาม

** เวลา 10.15–10.30 น. และ 14.15–14.30 น. รับประทานอาหารว่างและเครื่องดื่ม



แบบสอบถามความพึงพอใจการจัดฝึกอบรมเชิงปฏิบัติการ
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล

IRRISAT&MATLAB

วัตถุประสงค์: แบบสอบถามความพึงพอใจนี้จัดทำขึ้นเพื่อประเมินผลการจัดฝึกอบรมเชิงปฏิบัติการเพื่อนำเสนอ สาธิต และฝึกปฏิบัติการการใช้งาน Cloud-Based IrriSAT Application และการสร้างแบบจำลองสำหรับงานบริหารจัดการเชื่อมด้วยโปรแกรม MATLAB ภายใต้โครงการวิจัยเรื่อง “กลยุทธ์การปรับเปลี่ยนแนวทางการปฏิบัติการอ่างเก็บน้ำสำหรับพัฒนาการบริหารจัดการน้ำต้นทุนในระยะยาวของเขื่อนภูมิพล (ระยะที่ 1)”

12 พฤศจิกายน 2563

คำชี้แจง: โปรดทำเครื่องหมาย ✓ ลงในช่องที่ตรงกับความเป็นจริงมากที่สุด

ตอนที่ 1 ข้อมูลทั่วไปเกี่ยวกับผู้ตอบแบบสอบถาม ประเภท บุคลากรภายนอก นักศึกษา

ตอนที่ 2 ระดับความพึงพอใจ

หัวข้อประเมิน		ระดับความพึงพอใจ		
		ดี (3)	พอใช้ (2)	น้อย (1)
1.	วิทยากร			
	1.1 หัวข้อบรรยายสอดคล้องกับวัตถุประสงค์ของการจัดฝึกอบรม			
	1.2 วิทยากรนำเสนอและฝึกปฏิบัติเข้าใจง่ายและครบถ้วน			
2.	สถานที่และเวลา			
	2.1 สถานที่จัดฝึกอบรมเหมาะสม			
	2.2 อาหารและเครื่องดื่ม			
	2.3 เวลาในการจัดฝึกอบรมสอดคล้องกับเนื้อหา			
	2.4 มีความพร้อมด้านเอกสารประกอบการจัดฝึกอบรม			
3.	ประโยชน์ที่คาดว่าจะได้รับ			
	3.1 ท่านได้ประโยชน์จากการอบรมนี้			
	3.2 ท่านคาดว่าจะนำความรู้ความเข้าใจไปประยุกต์ใช้ในการปฏิบัติงานจริงได้			
	3.3 ท่านพึงพอใจการจัดฝึกอบรมนี้ในภาพรวม			

ข้อเสนอแนะเพิ่มเติม

.....

.....

.....

.....



**แบบสอบถามความพึงพอใจการจัดฝึกอบรมเชิงปฏิบัติการ
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล**

R & MINIZINC

วัตถุประสงค์: แบบสอบถามความพึงพอใจนี้จัดทำขึ้นเพื่อประเมินผลการจัดฝึกอบรมเชิงปฏิบัติการเพื่อนำเสนอ สาธิต และฝึกปฏิบัติการการบริหารจัดการเชื่อมด้วยเทคนิคปัญญาประดิษฐ์เบื้องต้นด้วยภาษา R และ MiniZinc ภายใต้โครงการวิจัยเรื่อง “กลยุทธ์การปรับเปลี่ยนแนวทางการปฏิบัติการอ่างเก็บน้ำสำหรับพัฒนาการบริหารจัดการน้ำต้นทุนในระยะยาวของเขื่อนภูมิพล (ระยะที่ 1)”

13 พฤศจิกายน 2563

คำชี้แจง: โปรดทำเครื่องหมาย ✓ ลงในช่องที่ตรงกับความเป็นจริงมากที่สุด

ตอนที่ 1 ข้อมูลทั่วไปเกี่ยวกับผู้ตอบแบบสอบถาม ประเภท บุคลากรภายนอก นักศึกษา

ตอนที่ 2 ระดับความพึงพอใจ

หัวข้อประเมิน		ระดับความพึงพอใจ		
		ดี (3)	พอใช้ (2)	น้อย (1)
1.	วิทยากร			
	1.1 หัวข้อบรรยายสอดคล้องกับวัตถุประสงค์ของการจัดฝึกอบรม			
	1.2 วิทยากรนำเสนอและฝึกปฏิบัติเข้าใจง่ายและครบถ้วน			
2.	สถานที่และเวลา			
	2.1 สถานที่จัดฝึกอบรมเหมาะสม			
	2.2 อาหารและเครื่องดื่ม			
	2.3 เวลาในการจัดฝึกอบรมสอดคล้องกับเนื้อหา			
	2.4 มีความพร้อมด้านเอกสารประกอบการจัดฝึกอบรม			
3.	ประโยชน์ที่คาดว่าจะได้รับ			
	3.1 ท่านได้ประโยชน์จากการอบรมนี้			
	3.2 ท่านคาดว่าจะนำความรู้ความเข้าใจไปประยุกต์ใช้ในการปฏิบัติงานจริงได้			
	3.3 ท่านพึงพอใจการจัดฝึกอบรมนี้ในภาพรวม			

ข้อเสนอแนะเพิ่มเติม

.....

.....

.....

.....



แผนงานที่ 3 การพัฒนาเทคโนโลยีสนับสนุนเพื่อการพัฒนาาระบบอัจฉริยะ

กลยุทธ์การปรับเปลี่ยนแนวทางการปฏิบัติการอ่างเก็บน้ำสำหรับพัฒนาการบริหารจัดการน้ำต้นทุนในระยะยาวของเขื่อนภูมิพล (ระยะที่ 1)
โครงการวิจัยเข้มมุ่ง สำนักประสานงานวิจัยการจัดการน้ำเชิงยุทธศาสตร์
สำนักงานคณะกรรมการส่งเสริมวิทยาศาสตร์ วิจัย และนวัตกรรม

รศ.ดร.อารีญา ฤทธิมา

ภาควิชาวิศวกรรมโยธาและสิ่งแวดล้อม คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล

การจัดฝึกอบรมเชิงปฏิบัติการ

E-mail: areeya.rit@mahidol.ac.th

12-13 พฤศจิกายน 2563



โครงการวิจัยเข็มมุ่ง สำนักประสานงานวิจัยการจัดการน้ำเชิงยุทธศาสตร์ สำนักงานคณะกรรมการส่งเสริมวิทยาศาสตร์ วิจัย และนวัตกรรม

รศ.ดร.สุจริต คุณธนกุลวงศ์

ประธานแผนงานยุทธศาสตร์เป้าหมาย โครงการวิจัยเข็มมุ่ง สำนักประสานงานวิจัยการจัดการน้ำ
เชิงยุทธศาสตร์ สกสว.

คณะนักวิจัย





คณะนักวิจัยมหาวิทยาลัยมหิดล-เกษตรศาสตร์



รศ.ดร.วราวุธ วุฒิวณิชย์ (ที่ปรึกษา)

ภาควิชาวิศวกรรมชลประทาน คณะวิศวกรรมศาสตร์ กำแพงแสน มหาวิทยาลัยเกษตรศาสตร์

E-mail : fengvwv@ku.ac.th



รศ.ดร.อารีญา ฤทธิมา (หัวหน้าโครงการวิจัย)

ภาควิชาวิศวกรรมโยธาและสิ่งแวดล้อม
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล

E-mail : areeya.rit@mahidol.ac.th



อ.ดร.ยุทธนา พันธุ์กมลศิลป์

สาขาวิชาวิศวกรรมสิ่งแวดล้อมและการจัดการภัยพิบัติ
มหาวิทยาลัยมหิดล

E-mail : yutthana.pha@mahidol.ac.th



อ.ดร.อรัญย์ ศรีรัตนทา ทาบุญานอน

คณะสิ่งแวดล้อมและทรัพยากรศาสตร์
มหาวิทยาลัยมหิดล

E-mail : allansriratana.tab@mahidol.ac.th



อ.ดร.วุฒิชชาติ แสงผล

คณะเทคโนโลยีสารสนเทศและการสื่อสาร
มหาวิทยาลัยมหิดล

E-mail : wudhichart.saw@mahidol.edu



อ.ดร.จิตดากา ไกรสังข์

คณะเทคโนโลยีสารสนเทศและการสื่อสาร
มหาวิทยาลัยมหิดล

E-mail : jidapa.kra@mahidol.edu



อ.ดร.ยุทธนา ตาละลักษมณ

ภาควิชาวิศวกรรมทรัพยากรน้ำ
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

E-mail : fengynt@ku.ac.th



แผนงานวิจัยที่ 3

การพัฒนาเทคโนโลยีเพื่อสนับสนุนการบริหารจัดการเขื่อน

กลุ่มงานที่ 3.1

ศูนย์ข้อมูลแผนงานการบริหารจัดการน้ำ

กลุ่มงานที่ 3.2

การศึกษาและพัฒนาระบบการใช้ระบบตรวจจับพื้นที่สีเขียว พร้อมระบบสารสนเทศ

ระบบวิเคราะห์ข้อมูลขนาดใหญ่เพื่อการวางแผนงานการบริหารจัดการน้ำ

การอบรมการวิเคราะห์ข้อมูลขนาดใหญ่เพื่อการวางแผนบริหารจัดการน้ำ

การพัฒนาระบบคาดการณ์ปริมาณฝนสองสัปดาห์เพื่อการบริหารจัดการน้ำ

กลุ่มงานที่ 3.3

กลยุทธ์การปรับเปลี่ยนแนวทางปฏิบัติการอ่างเก็บน้ำสำหรับบริหารจัดการน้ำต้นฤดูระยะยาวของเขื่อนภูมิพล (ระยะที่ 1)

การประเมินปริมาณความต้องการน้ำในพื้นที่ราบภาคกลาง (ระยะที่ 1)

การศึกษาและประเมินปริมาณน้ำต้นทุน (น้ำท่า น้ำผิวดิน น้ำบาดาล) ในพื้นที่ลุ่มน้ำเจ้าพระยาตอนล่าง

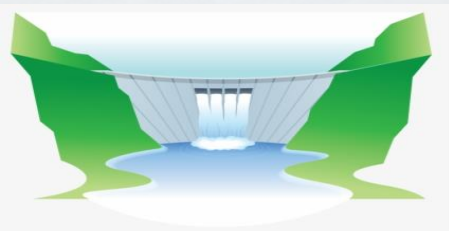
กลุ่มงานที่ 3.4

Chao Phraya Delta 2020

การประเมินความเสี่ยงของน้ำท่วมและน้ำแล้ง

เป้าหมายหลักของแผนงานที่ 3.3

- คาดการณ์ปริมาณฝนสองสัปดาห์
- ประเมินปริมาณความต้องการน้ำในพื้นที่โครงการลุ่มน้ำเจ้าพระยาใหญ่
- ศึกษาและประเมินปริมาณน้ำต้นทุน (น้ำท่า น้ำผิวดิน และน้ำบาดาล) ในพื้นที่ลุ่มน้ำเจ้าพระยาตอนล่างโดยประยุกต์ใช้แบบจำลอง DWCM-AgWU
- ทดสอบกลยุทธ์ในการบริหารเขื่อนที่จะเพิ่มปริมาณเก็บกักของอ่างเก็บน้ำเขื่อนภูมิพลให้สูงขึ้น 15% ในช่วงต้นฤดูแล้งจากฐานข้อมูลในอดีตระยะยาว



เป้าหมายของงานบริหารเขื่อน :
เพิ่มปริมาณเก็บกักของอ่างเก็บน้ำเขื่อนภูมิพลให้สูงขึ้น 15% ในช่วงต้นฤดูแล้ง
(เดือนพฤศจิกายน) จากฐานข้อมูลในอดีตระยะยาว



กลยุทธ์ที่ 1



กลยุทธ์ที่ 2



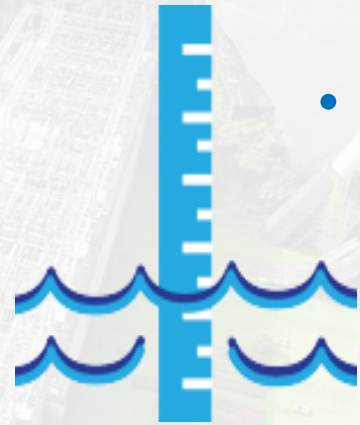
กลยุทธ์ที่ 3

Artificial Intelligence

- การปฏิบัติการอ่างเก็บน้ำรูปแบบใหม่ด้วยเทคนิค AI
 - FUZZY MODEL & ANFIS & RL
 - CP MODEL & ML



- การปรับลดพื้นที่ชลประทานในโครงการชลประทานเจ้าพระยาใหญ่ตามประเภทปีน้ำ



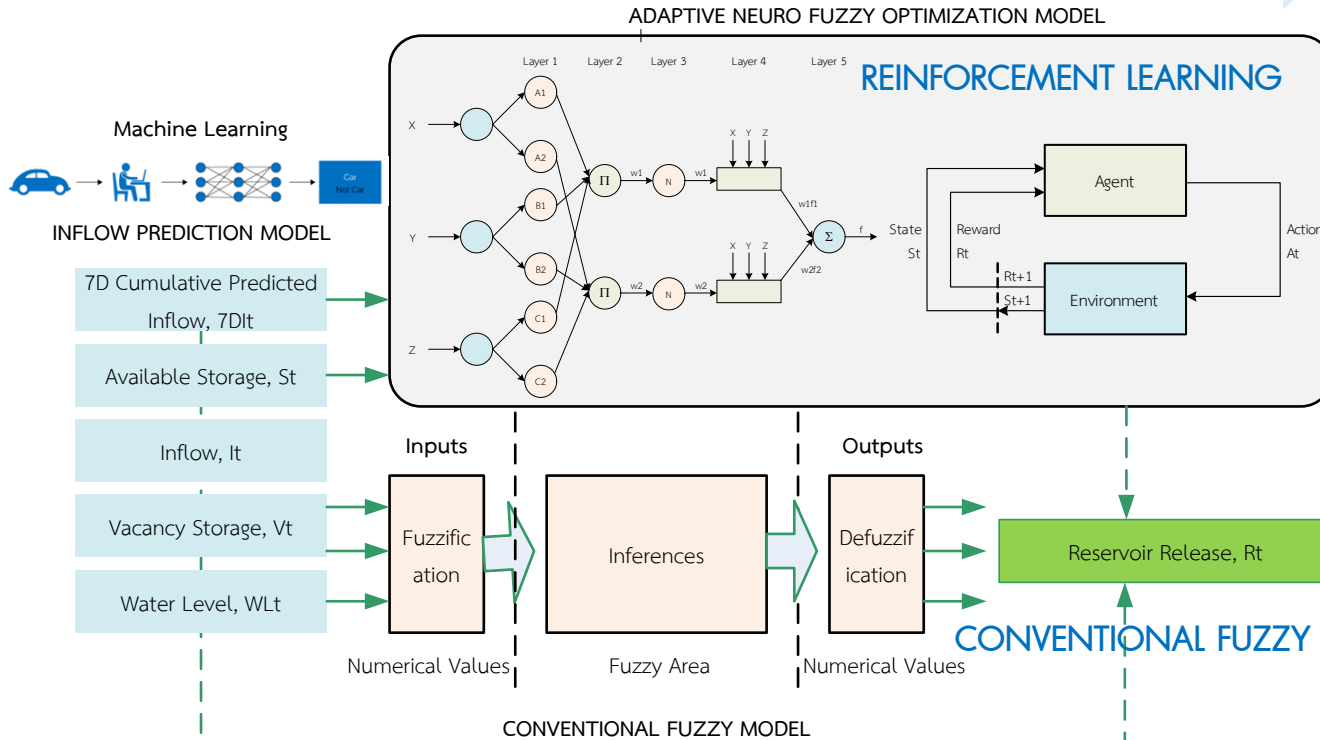
- การพิจารณา Sideflow ของสถานีหลักท้ายเขื่อนมาช่วยในการกำหนดการระบายน้ำ
 - ข้อมูลตรวจวัด
 - ข้อมูลจากแบบจำลอง DWCM-AgWU

กลยุทธ์ที่ 1 นำเสนอการปฏิบัติการอ่างเก็บน้ำรูปแบบใหม่ด้วยเทคนิค AI



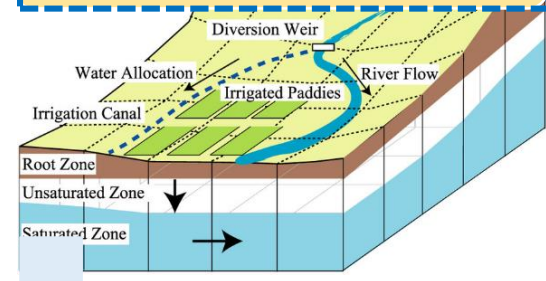
Predicted Rainfall at Lead time
14 Days (Dr.Kanoksri)

INFLOW PREDICTION MODEL
WITH MACHINE LEARNING



1 CONVENTIONAL FUZZY MODEL & REINFORCEMENT LEARNING & ADAPTIVE NEURO FUZZY OPTIMIZATION MODEL (ANFIS)

Satellite-Based Agricultural Water Demand (Dr.ChuPhan)
Localized Flow Downstream of BB Dam (W.4A & CT.2A) (Dr.Chaiyapong)



2 CONSTRAINT PROGRAMMING MODEL (CP)

```

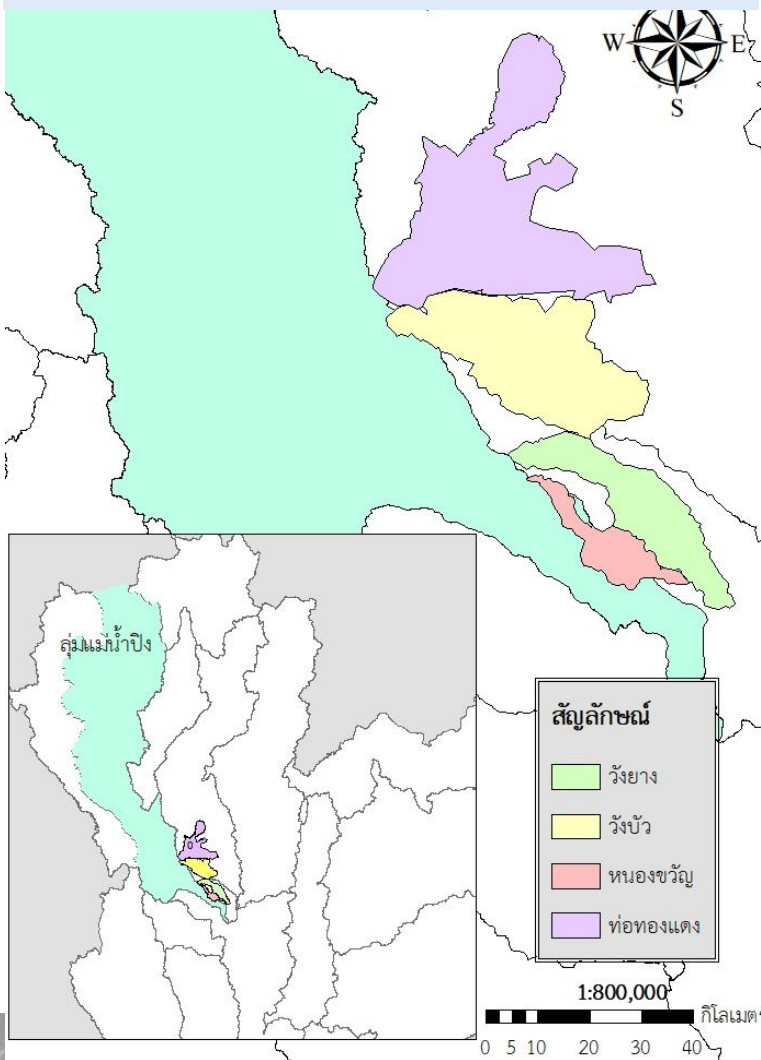
35 constraint forall(i in N){
36   St[i] = St[i] + It[i] - Et[i] - Rt[i]
37 };
38
39 constraint forall(i in N){
40   Smin <= St[i] & St[i] <= Smax
41 };
42
43 constraint forall(i in N){
44   Rmin <= Rt[i] & Rt[i] <= Rmax
45 };
46
47 constraint forall(i in N){
48   Gmin <= Gt[i] & Gt[i] <= Gmax
49 };
50

```

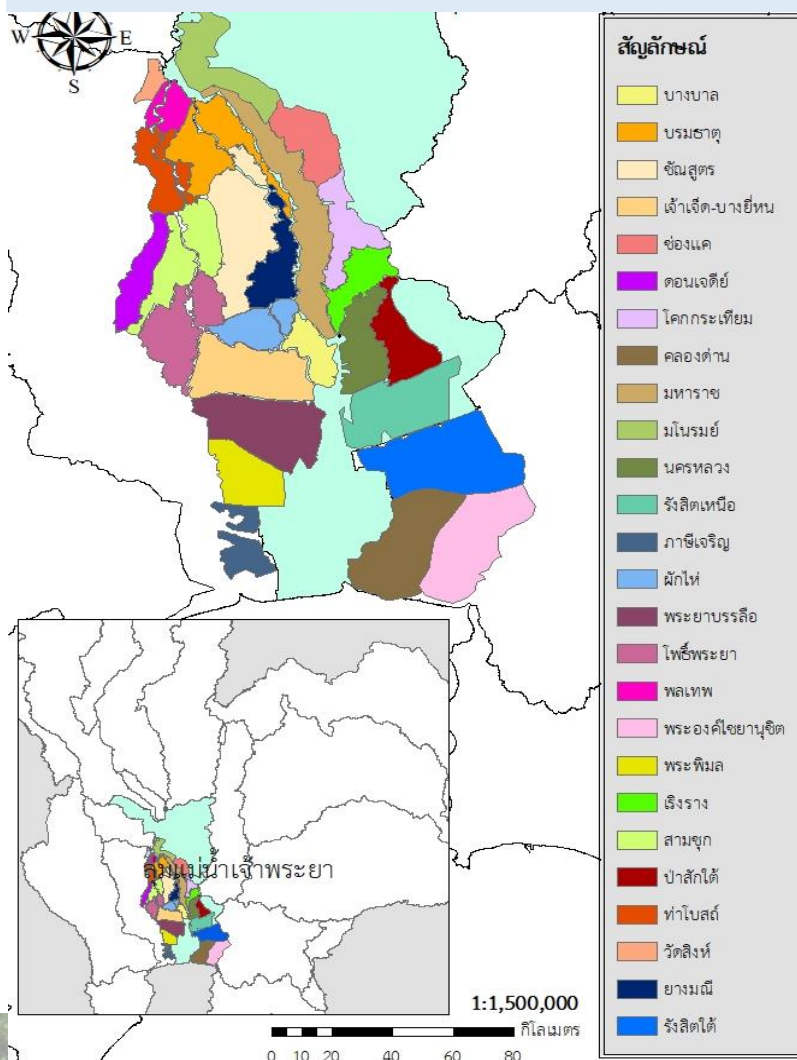
STOCHASTIC CONSTRAINT PROGRAMMING MODEL

กลยุทธ์ที่ 2 การปรับลดพื้นที่ชลประทานในโครงการชลประทานเจ้าพระยาใหญ่ตามประเภทปีน้ำ

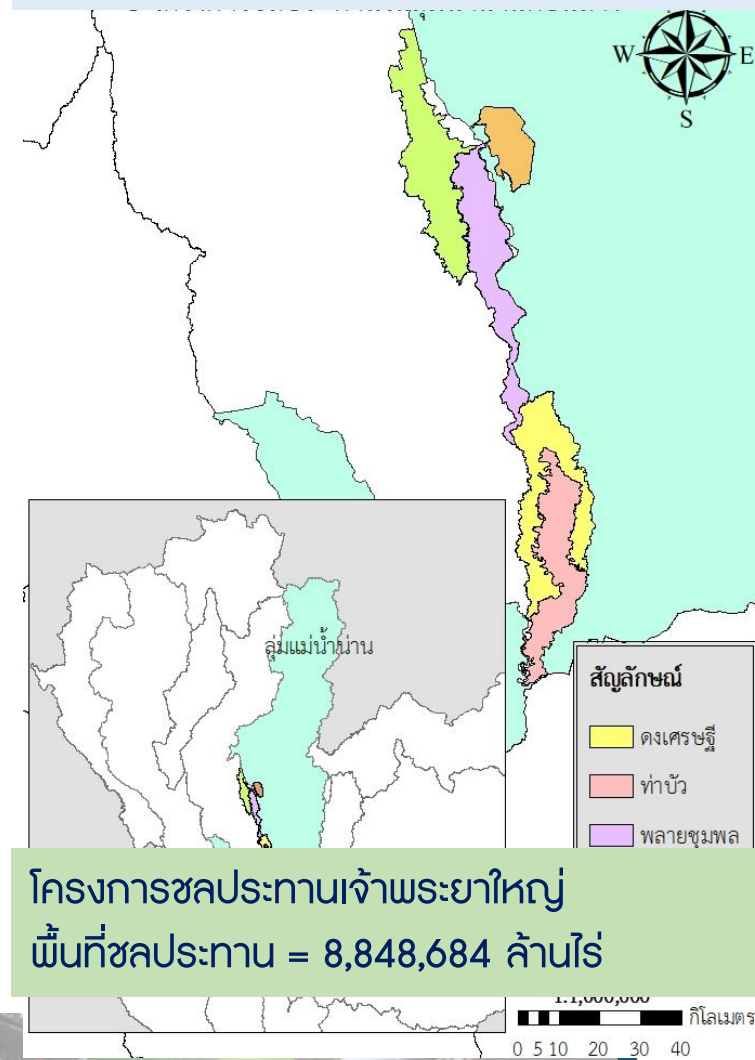
4 โครงการชลประทานในกลุ่มน้ำปิงตอนล่าง



26 โครงการชลประทานในกลุ่มน้ำเจ้าพระยา-ท่าจีน



5 โครงการชลประทานในกลุ่มน้ำน่านตอนล่าง



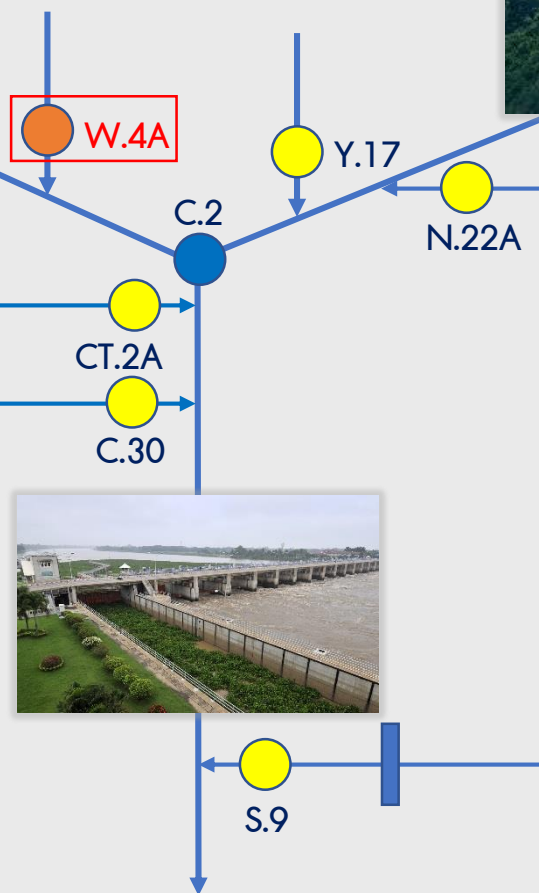
กลยุทธ์ที่ 2 การปรับลดพื้นที่ชลประทานในโครงการชลประทานเจ้าพระยาใหญ่ตามประเภทปีน้ำ

แหล่งข้อมูล	ปริมาณความต้องการน้ำ (ล้านลูกบาศก์เมตรต่อปี)				ผลต่าง
	การอุปโภคบริโภคและการท่องเที่ยว ^{1/ & 2/}	การอุตสาหกรรม ^{3/}	การเกษตรกรรม ^{4/}	ระบบนิเวศ ^{5/}	
โครงการพัฒนาระบบคลังข้อมูล 25 กลุ่มน้ำ (สสน., 2555)	236.42 (พ.ศ. 2551)	931.95 (พ.ศ. 2547)	7,787.60 (พ.ศ. 2546)	2,386.41 (11,342.38)	-4,477.31
โครงการศึกษาความมั่นคงของกลุ่มน้ำอย่างยั่งยืนทั้ง 25 กลุ่มน้ำ (กรมทรัพยากรน้ำ, 2559)	1,428.79 (พ.ศ. 2558)	442.97 (พ.ศ. 2558)	6,355 (พ.ศ. 2556)	3,156.12 (11,382.88)	-4,436.81
โครงการวิจัย ดร.อาริยา (สกสว., 2563)	518.31 (พ.ศ. 2561)	1,187.71 (พ.ศ. 2561)	9,984.98 (พ.ศ. 2561)	4,128.69 (15,819.69)	-
โครงการวิจัย ดร.ชูพันธุ์ (สกสว., 2563)	790.70 (พ.ศ. 2561)	1,618 (พ.ศ. 2561)	9,298 (พ.ศ. 2561)	-	ปริมาณความต้องการน้ำภาคเกษตรกรรมเพิ่มสูงขึ้น

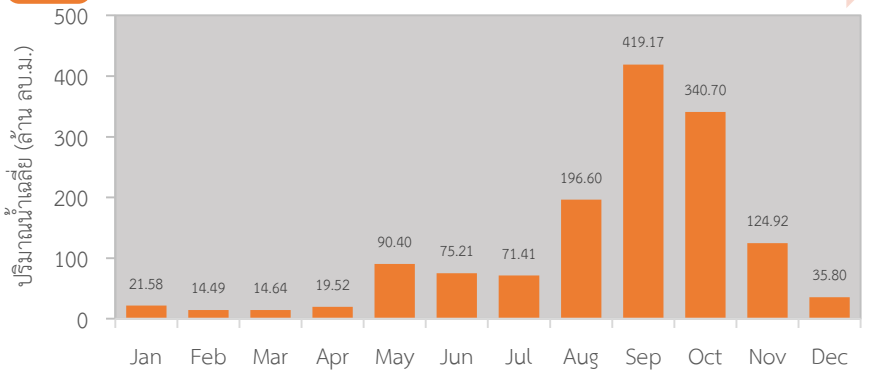
● ลดปริมาณความต้องการน้ำเพื่อการเกษตรกรรม 1,700 ล้าน ลบ.ม. ต่อปี

พื้นที่เพาะปลูกที่แนะนำ	ปีน้ำมาก	ปีน้ำปกติ	ปีน้ำน้อย
ฤดูฝน (ล้านไร่)	7	6	5
ฤดูแล้ง (ล้านไร่)	5	4	2

กลยุทธ์ที่ 3 การพิจารณา Sideflow ของสถานีหลักท้ายเขื่อนมาช่วยในการกำหนดการระบายน้ำ



● ปริมาณน้ำท่ารายปีของสถานี W.4A
1,424 ล้าน ลบ.ม. ต่อปี



กลยุทธ์	ผลการจำลองระบบ	ช่วงการจำลองระบบ	เปอร์เซ็นต์ปริมาณน้ำเก็บกักที่เพิ่มขึ้น/ลดลง (%Active Storage) ^{1/}			
			พฤศจิกายน	ฤดูฝน	ฤดูแล้ง	รายปี
แนวทางที่ 1 Conventional Fuzzy Logic Model สถานการณ์ในอดีตถึงปัจจุบัน						
กลยุทธ์ 1	กรณีกำหนดปริมาณความต้องการน้ำเป้าหมายตามแผนการจัดสรรน้ำของ กฟผ. (สภาพจริง)	2543–2561	+6.09	+18.37	+11.57	+14.70
กลยุทธ์ 2	กรณีปรับลดพื้นที่เพาะปลูกของโครงการเจ้าพระยาใหญ่ตั้งแต่ปี พ.ศ. 2555–2561	2543–2561	+9.86 ($\Delta 3.77$) ^{2/}	+24.50 ($\Delta 6.13$) ^{2/}	+16.13 ($\Delta 4.56$) ^{2/}	+19.98 ($\Delta 5.28$) ^{2/}
กลยุทธ์ 3	กรณีพิจารณาปริมาณ Sideslow สถานี W.4A ในการกำหนดการระบายน้ำจากเขื่อนภูมิพล & กำหนดปริมาณความต้องการน้ำเป้าหมายตามแผนการจัดสรรน้ำของ กฟผ. ^{3/}	2543–2561	+14.55 ($\Delta 8.46$) ^{2/}	+25.69 ($\Delta 7.32$) ^{2/}	+19.37 ($\Delta 7.80$) ^{2/}	+22.28 ($\Delta 7.58$) ^{2/}

สามารถบรรลุเป้าหมายในการเพิ่มน้ำต้นทุน 15% ในช่วงฤดูแล้ง

หมายเหตุ : ^{1/} เปรียบเทียบกับปริมาณน้ำเก็บกักจริงระหว่างปี พ.ศ. 2543–2561

^{2/} ผลต่างคำนวณจากการเปรียบเทียบกับกรณี 1

^{3/} ยังเกิดการไหลล้นอ่างเก็บน้ำในปี พ.ศ. 2554

กลยุทธ์	ผลการจำลองระบบ	ช่วงการจำลองระบบ	เปอร์เซ็นต์ปริมาณน้ำเก็บกักที่เพิ่มขึ้น/ลดลง (%Active Storage) ^{1/}			
			พฤศจิกายน	ฤดูฝน	ฤดูแล้ง	รายปี
แนวทางที่ 2 Constraint Programming with ML สถานการณ์ในอดีตถึงปัจจุบัน						
กลยุทธ์ 1	กรณีกำหนดสมการข้อจำกัดรายปี & ปริมาณความต้องการน้ำเป้าหมายตามแผนการจัดสรรน้ำของ กฟผ. (สภาพจริง) ^{3/}	2543–2561	+7.94	+14.00	+10.36	+12.03
กลยุทธ์ 2	กรณีกำหนดสมการข้อจำกัดรายฤดูกาล & ปริมาณความต้องการน้ำเป้าหมายตามแผนการจัดสรรน้ำของ กฟผ. (สภาพจริง) ^{3/}	2543–2561	+7.10 ($\Delta-0.84$) ^{2/}	+12.93 ($\Delta-1.07$) ^{2/}	+9.41 ($\Delta-0.95$) ^{2/}	+11.03 ($\Delta-1.00$) ^{2/}
กลยุทธ์ 3	กรณีกำหนดสมการข้อจำกัดรายฤดูกาล & ปริมาณความต้องการน้ำเป้าหมายตามแผนการจัดสรรน้ำของ กฟผ. (สภาพจริง) & พิจารณาปริมาณ Sideflow สถานี W.4A ^{3/}	2543–2561	+10.49 ($\Delta 2.55$) ^{2/}	+17.67 ($\Delta 3.67$) ^{2/}	+13.12 ($\Delta 2.76$) ^{2/}	+15.21 ($\Delta 3.18$) ^{2/}

สามารถบรรลุเป้าหมายในการเพิ่มน้ำต้นทุน 15% ในช่วงฤดูแล้ง

หมายเหตุ : ^{1/} เปรียบเทียบกับปริมาณน้ำเก็บกักจริงระหว่างปี พ.ศ. 2543–2561

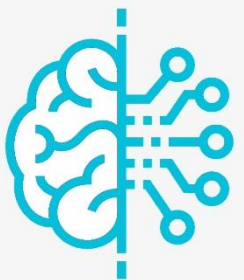
^{2/} ผลต่างคำนวณจากการเปรียบเทียบกับกรณี 1 ^{3/} ยังเกิดการไหลล้นอ่างเก็บน้ำในปี พ.ศ. 2554

แผนงานที่ 3 การพัฒนาเทคโนโลยีสนับสนุนเพื่อการพัฒนาาระบบอัจฉริยะ:

กลยุทธ์การปรับเปลี่ยนแนวทางการปฏิบัติการอ่างเก็บน้ำสำหรับพัฒนาการบริหารจัดการน้ำต้นทุนในระยะยาวของเขื่อนภูมิพล (ระยะที่ 1)

เปรียบเทียบกับปริมาณน้ำเก็บกักจริงระหว่างปี พ.ศ. 2543-2561

สามารถบรรลุเป้าหมายในการเพิ่มน้ำต้นทุน 15% ในช่วงฤดูแล้ง



Artificial Intelligence

- การปฏิบัติการอ่างเก็บน้ำรูปแบบใหม่ด้วยเทคนิค AI
 - FUZZY MODEL & ANFIS & RL
 - CP MODEL & ML



- การปรับลดพื้นที่ชลประทานในโครงการชลประทานเจ้าพระยาใหญ่ตามประเภทปีน้ำ

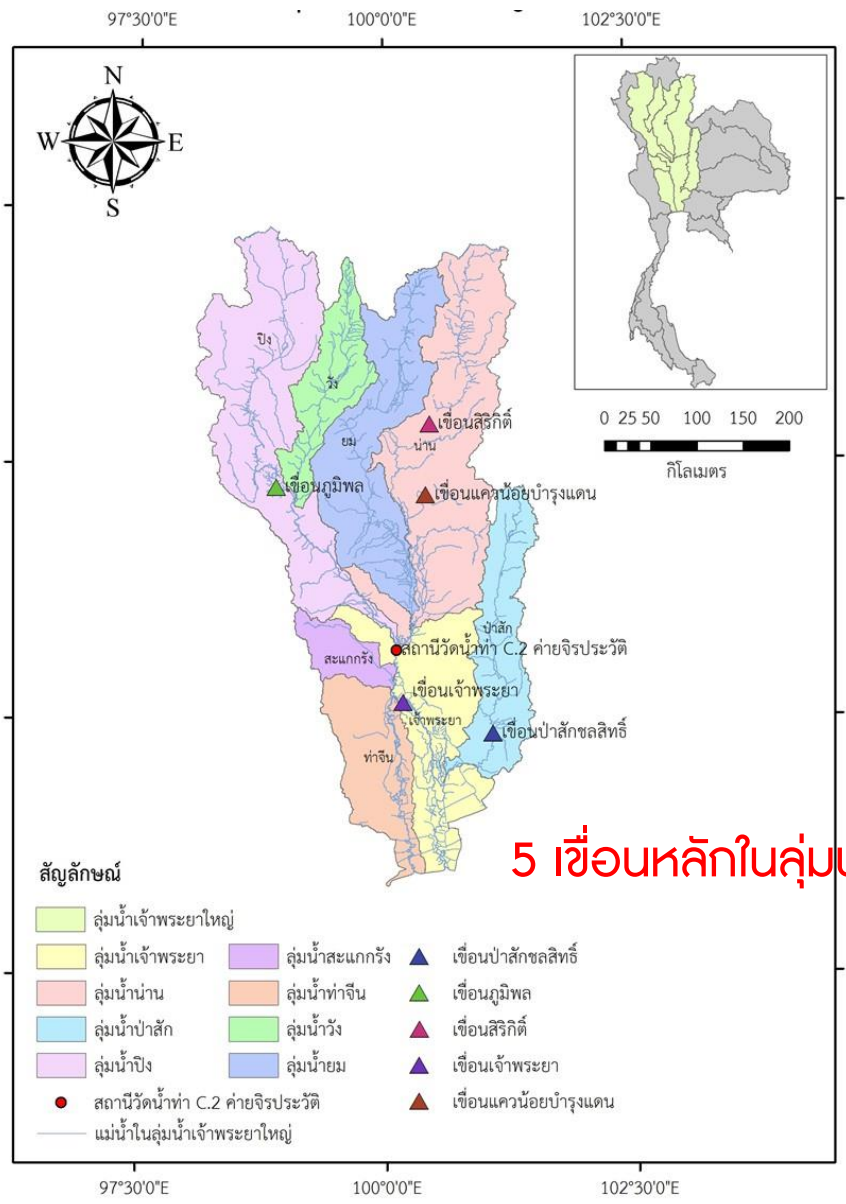


- การพิจารณา Sideflow ของสถานีหลักท้ายเขื่อนมาช่วยในการกำหนดการระบายน้ำ
 - ข้อมูลตรวจวัด
 - ข้อมูลจากแบบจำลอง DWCM-AgWU

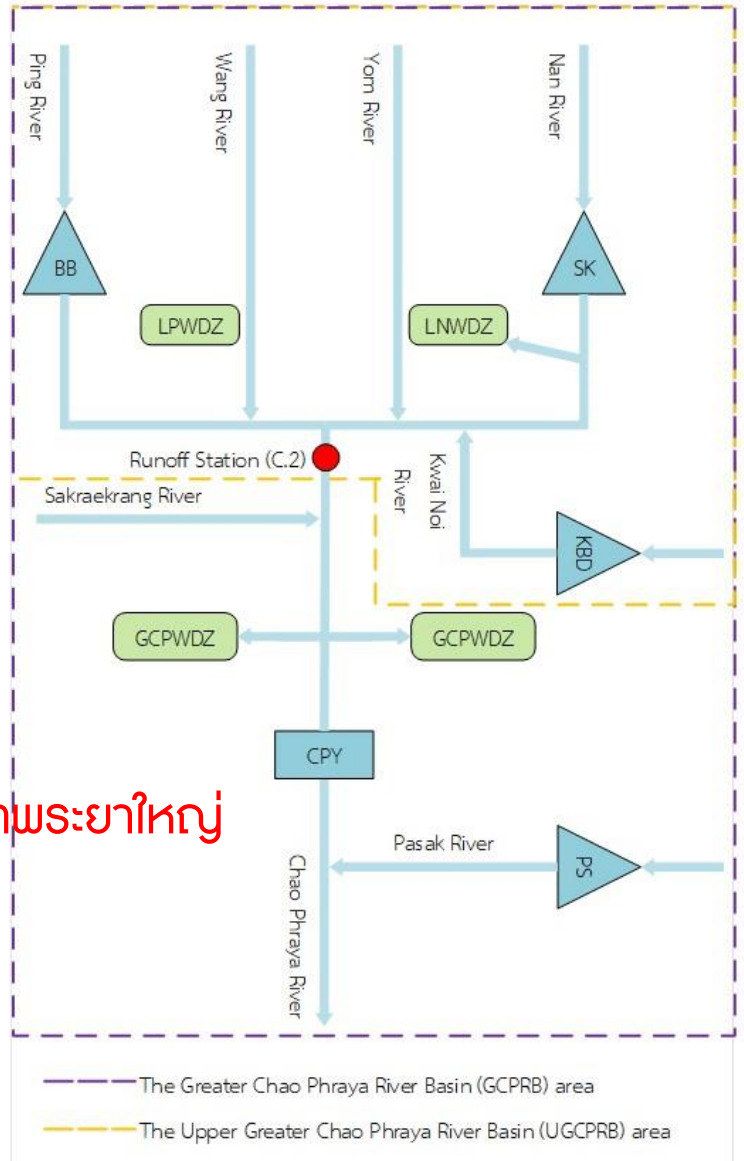


แผนงานในอนาคต

การปฏิบัติการระบบอ่างเก็บน้ำรูปแบบใหม่สำหรับการบริหารจัดการน้ำต้นทุนระยะยาว ในกลุ่มน้ำเจ้าพระยาใหญ่ด้วยเทคนิคปัญญาประดิษฐ์ (ระยะที่ 2)



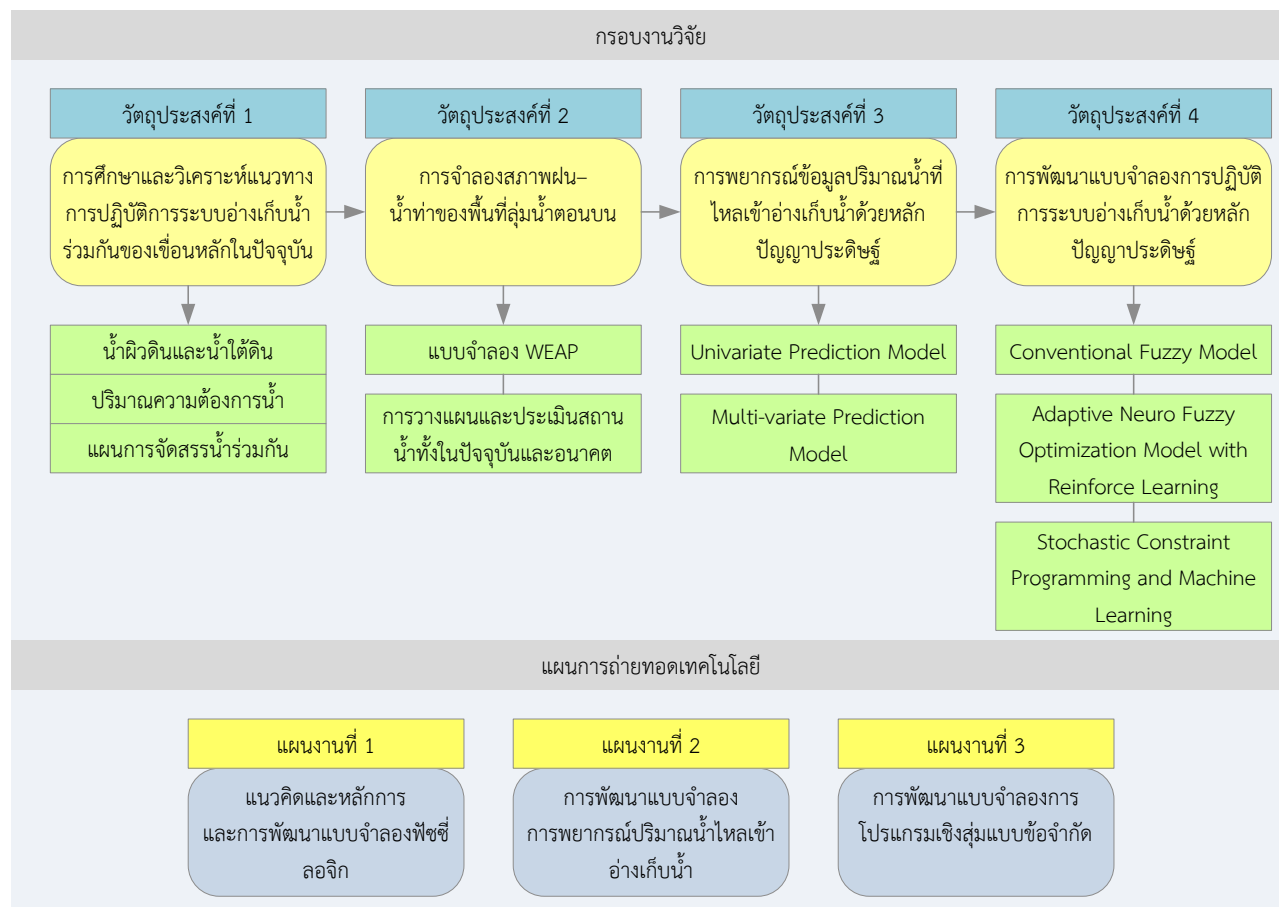
5 เขื่อนหลักในกลุ่มน้ำเจ้าพระยาใหญ่



เป้าหมายหลัก : เพิ่มน้ำต้นทุน & แก้ปัญหา น้ำท่วม-ภัยแล้ง

- การปฏิบัติการอ่างเก็บน้ำแบบหลายอ่าง (Multi-Reservoir Operation System)
- การปฏิบัติการระบบอ่างเก็บน้ำรูปแบบใหม่ด้วยเทคนิค AI
- FUZZY MODEL & ANFIS & RL
- CP MODEL & ML

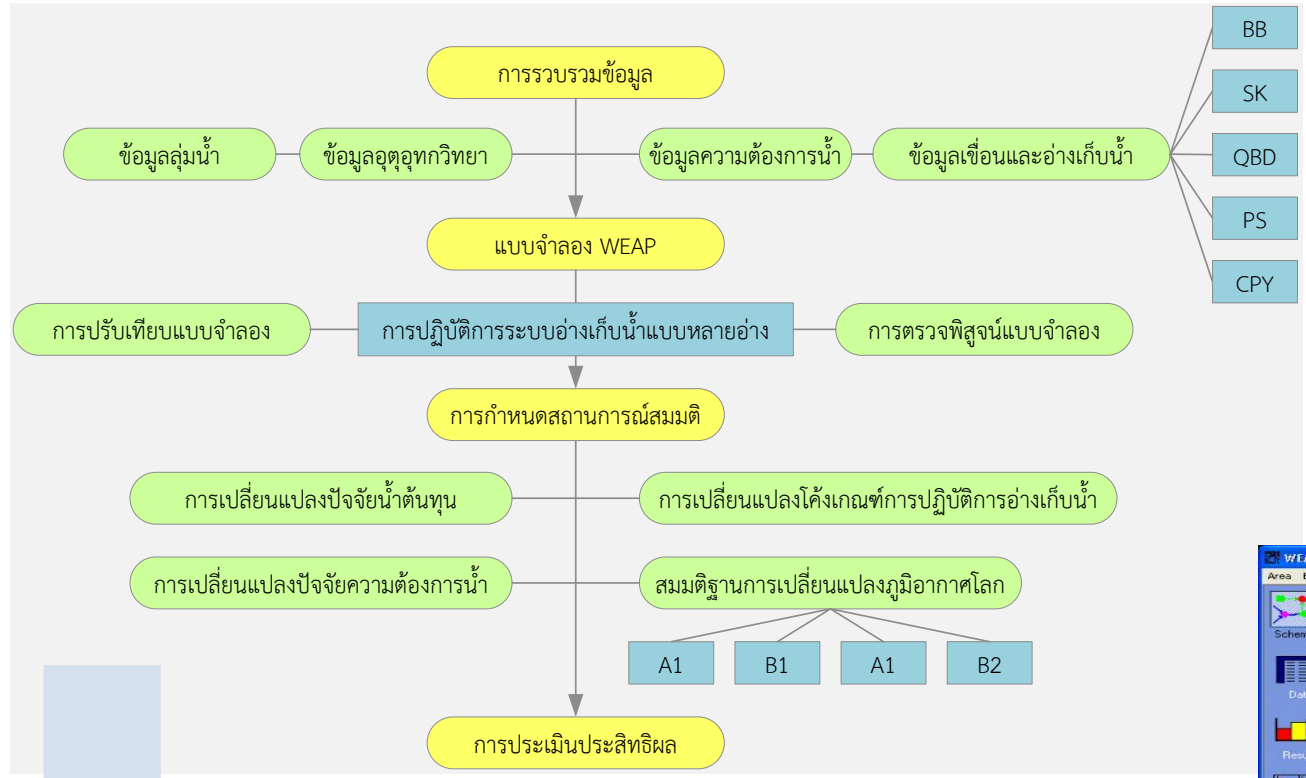
กรอบงานวิจัย



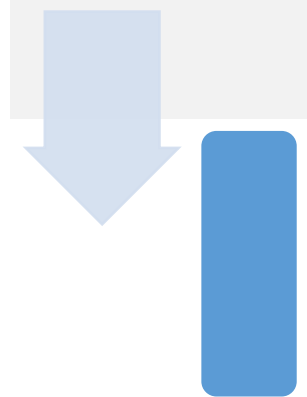
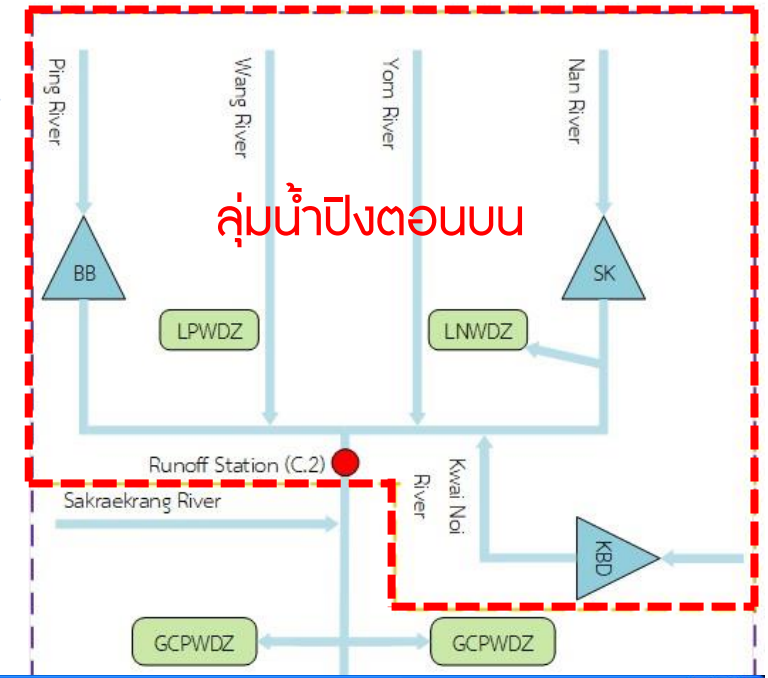
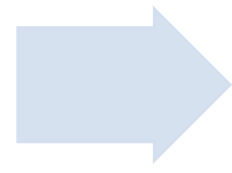
วัตถุประสงค์ของงานวิจัย

- 1 ● Analysis of Multi-Reservoir Operation System
- 2 ● Watershed System Modelling
- 3 ● Reservoir Inflow Forecast
- 4 ● Multi-Reservoir Re-Operation Using Artificial Intelligence

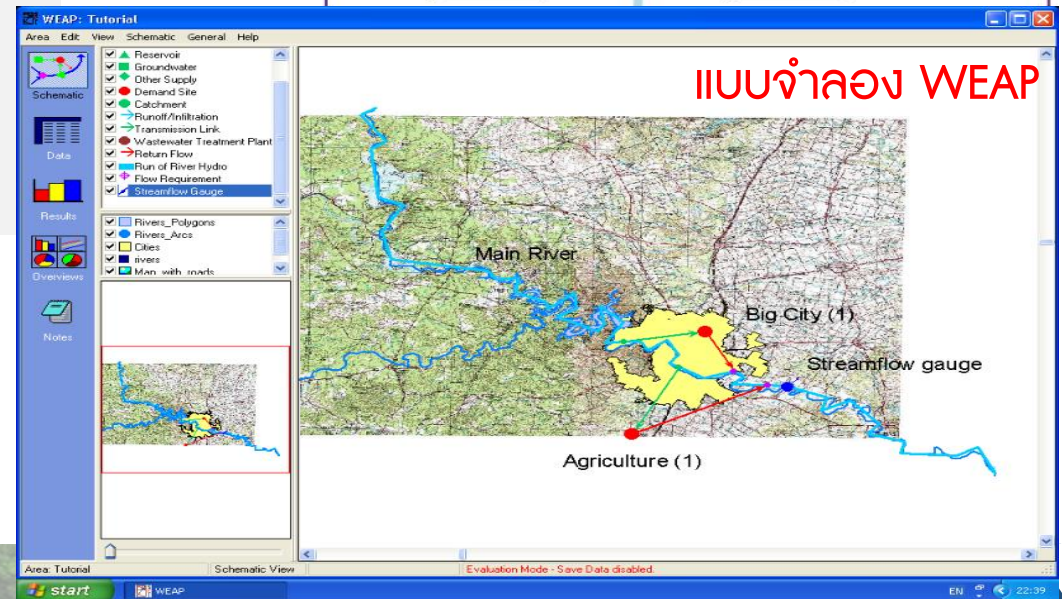
การจำลองสภาพฝน-น้ำท่าในพื้นที่ลุ่มน้ำปิงตอนบน



- BB
- SK
- QBD
- PS
- CPY



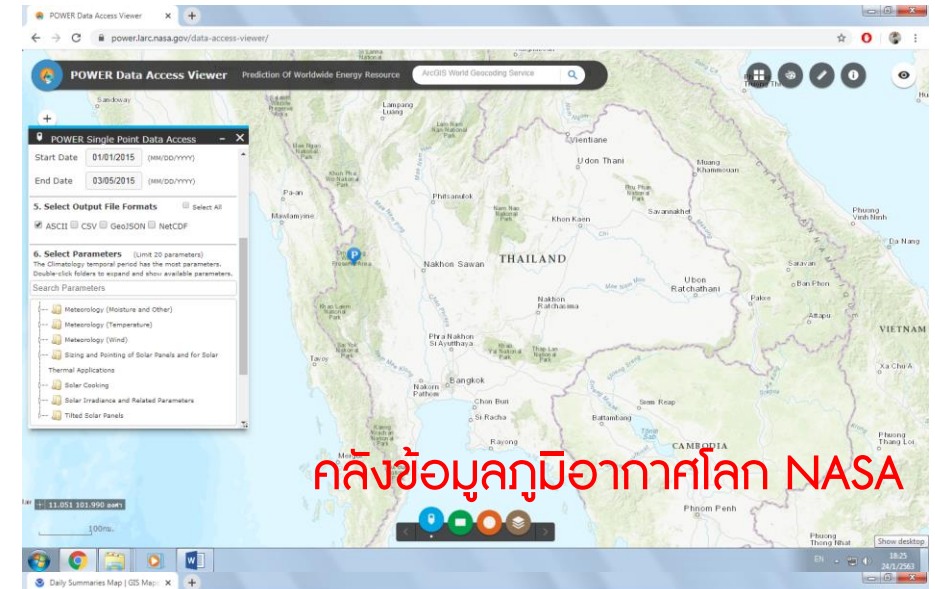
● กำหนดแผนการจัดสรรน้ำจากผลลัพธ์ของแบบจำลอง WEAP ให้มีความเหมาะสมต่อสถานการณ์น้ำรูปแบบต่าง ๆ



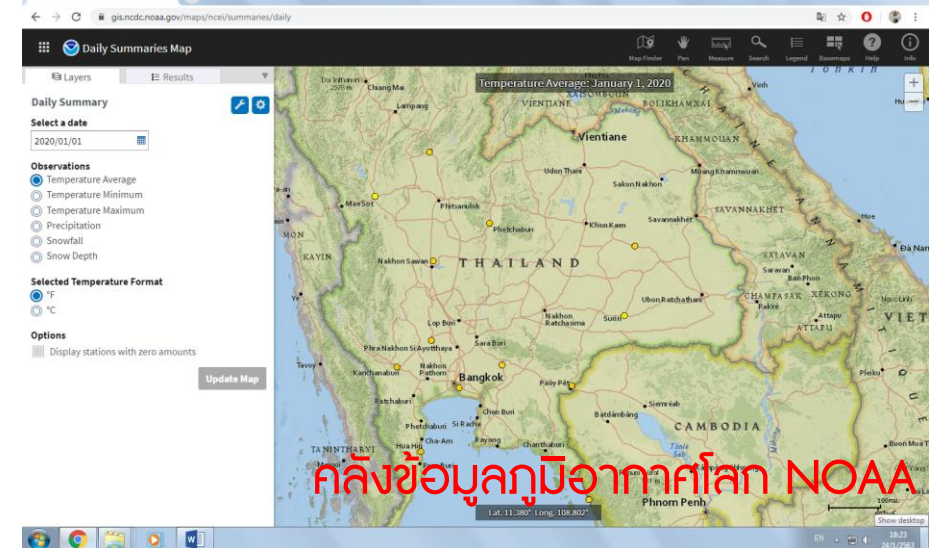
การพยากรณ์ปริมาณน้ำไหลเข้าอ่างเก็บน้ำด้วยหลักปัญญาประดิษฐ์

- ส่วนที่ 1 ข้อมูลผลการพยากรณ์ปริมาณน้ำฝนล่วงหน้า (Predicted Precipitation Data) รายวันและรายฤดูกาลจาก 2 หน่วยงานหลักในประเทศไทย ได้แก่ กรมอุตุนิยมวิทยาและ/หรือสถาบันสารสนเทศทรัพยากรน้ำ (องค์การมหาชน) ซึ่งมีลักษณะเป็นข้อมูลฝนพยากรณ์เชิงพื้นที่

- ส่วนที่ 2 คลังข้อมูลสภาพภูมิอากาศโลก (Global Climate Data) จาก 2 องค์การหลัก ได้แก่ องค์การบริหารการบินและอวกาศแห่งชาติ (National Aeronautics and Space Administration, NASA) ซึ่งเป็นหน่วยงานที่รับผิดชอบในโครงการอวกาศและงานวิจัยห้วงอวกาศของสหรัฐอเมริกา และองค์การบริหารมหาสมุทรและชั้นบรรยากาศแห่งชาติ (National Oceanic and Atmospheric Administration, NOAA) เป็นหน่วยงานทางวิทยาศาสตร์ของสหรัฐอเมริกาในกระทรวงพาณิชย์ของสหรัฐอเมริกา



คลังข้อมูลภูมิอากาศโลก NASA



คลังข้อมูลภูมิอากาศโลก NOAA

การปฏิบัติการระบบอ่างเก็บน้ำรูปแบบใหม่ด้วยเทคนิค AI

● FUZZY MODEL & ANFIS & RL

● CP MODEL & ML

รูปแบบของแบบจำลอง	ระบบอ่างเก็บน้ำทุกอ่างในพื้นที่ศึกษา
ประเภทของแบบจำลองการหาค่าที่ดีที่สุด	การหาค่าที่ดีที่สุดแบบหลายวัตถุประสงค์ (Multi-Objective Optimization)
ตัวแปรตัดสินใจ (Decision Variable)	ปริมาณการระบายน้ำที่เหมาะสมรายวันของแต่ละอ่างเก็บน้ำ (Optimum Daily Release)
ฟังก์ชันวัตถุประสงค์ (Objective Function)	<p>(1) ฟังก์ชันวัตถุประสงค์ในการตอบสนองความต้องการน้ำเพื่อกิจกรรมต่าง ๆ ทางด้านท้ายเขื่อนโดยลดปัญหาการขาดแคลนน้ำให้เหลือน้อยที่สุด (Minimization of Water Demand Deficit) :</p> <p>(2) ฟังก์ชันวัตถุประสงค์ในการผลิตพลังงานไฟฟ้าให้ได้สูงสุด (Maximization of Hydropower Generation)</p> <p>(3) ฟังก์ชันวัตถุประสงค์ในการเพิ่มปริมาณน้ำเก็บกักของอ่างเก็บน้ำด้วยแนวคิดการลดการปล่อยน้ำส่วนเกินในช่วงน้ำหลาก (Minimization of Surplus Release during Refilled Period)</p>

รูปแบบของแบบจำลอง	อ่างเก็บน้ำ 1	อ่างเก็บน้ำ 2	อ่างเก็บน้ำ 3	อ่างเก็บน้ำ 4
ตัวแปรนำเข้า (Input Variables)				
ตัวแปรนำเข้าในช่วงฤดูแล้ง :				
-ปริมาณน้ำเก็บกักเริ่มต้น (Available Storage)	√	√	√	√
-ปริมาณน้ำไหลเข้าอ่างเก็บน้ำ (Inflow)	√	√	√	√
-ปริมาณน้ำไหลเข้าด้านข้าง (Side Flow)	√	√	√	√
ตัวแปรนำเข้าในช่วงฤดูฝน :				
-ปริมาณน้ำไหลเข้าอ่างเก็บน้ำพยากรณ์ (Predicted Inflow)	√	√	√	√
-ปริมาตรว่างของอ่างเก็บน้ำ (Vacancy Storage)	√	√	√	√
-ระดับน้ำ (Water State/Level)	√	√	√	√
ตัวแปรผลลัพธ์ (Output Variables)				
ตัวแปรผลลัพธ์ในช่วงฤดูแล้ง :				
-สัดส่วนการระบายน้ำของแต่ละอ่างเก็บน้ำซึ่งสัมพันธ์กับปริมาณความต้องการน้ำ (Release Fraction-Water Demand)	√	√	√	√
ตัวแปรผลลัพธ์ในช่วงฤดูฝน :				
-สัดส่วนการระบายน้ำของแต่ละอ่างเก็บน้ำซึ่งสัมพันธ์กับปริมาณน้ำที่ไหลเข้าอ่างเก็บน้ำ (Release Fraction-Inflow)	√	√	√	√
ฟังก์ชันสมาชิก (Membership Function)	Piece-wise linear membership function [0-1]			
เทคนิคการอนุมาน (Inference Technique)	Mamdani inference mechanism			
วิธี Defuzzification	Centre of gravity method			



ขอขอบคุณ



Cloud-Based IrriSAT Application

Satellite Irrigation Advisory Service and Irrigation Monitoring. IrriSAT helps farmers and water managers in day-by-day work

Advisory irrigation system compliant with Italian DM MIPAAF 31 luglio 2015

Ask for Demo

Access



<https://www.irrisat.com/en/home-2>

อาจารย์ ดร.ยุทธนา พันธุ์มลศิลป์
สาขาวิชาวิศวกรรมสิ่งแวดล้อมและการจัดการภัยพิบัติ
มหาวิทยาลัยมหิดล วิทยาเขตกาญจนบุรี
12 พฤศจิกายน 2563

<https://irrisat-cloud.appspot.com/>

IrrisAT - Weather based irrigation scheduling

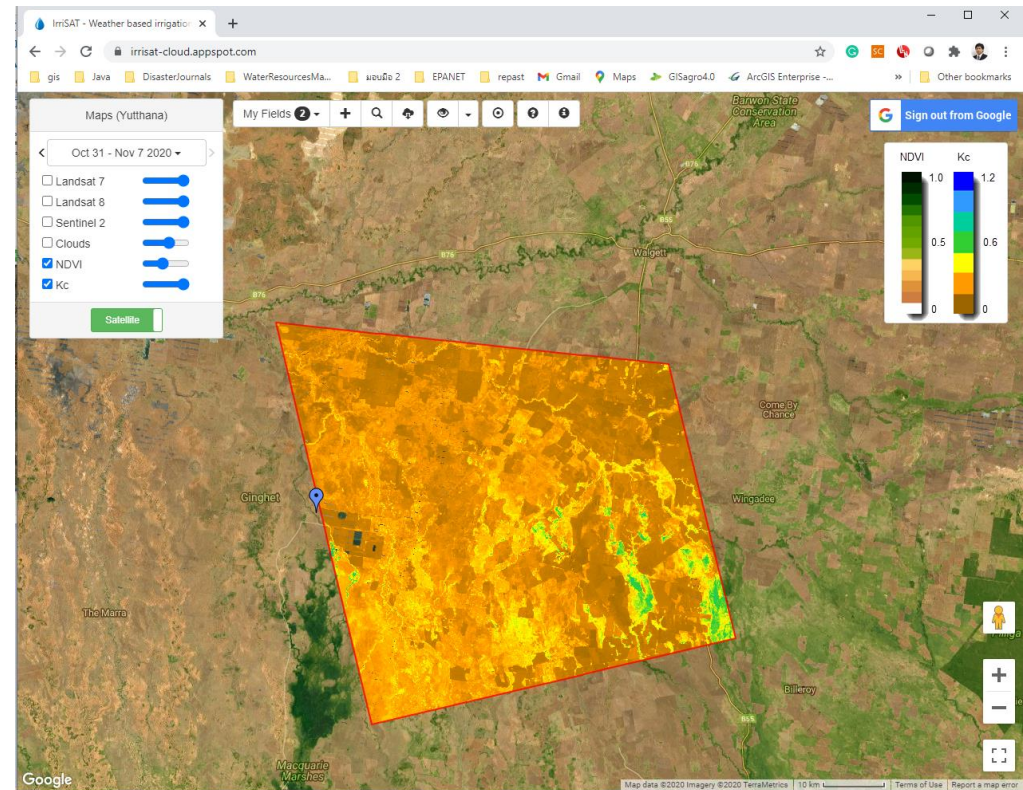
Realisation



Our Partners



Close



Documentation

IrriSAT - Quick Guide (2 MB [pdf])

IrriSAT Technical Reference (1 MB [pdf])

IrriSAT API Reference

Multimedia



September 2016

IrriSAT Technical Reference

John Hornbuckle¹, Jamie Vleeshouwer², Carlos Ballester¹, Janelle Montgomery³, Robert Hoogers³ & Robert Bridgart²

¹ Deakin University, Centre for Regional and Rural Futures

² CSIRO Land & Water

³ NSW Department of Primary Industries

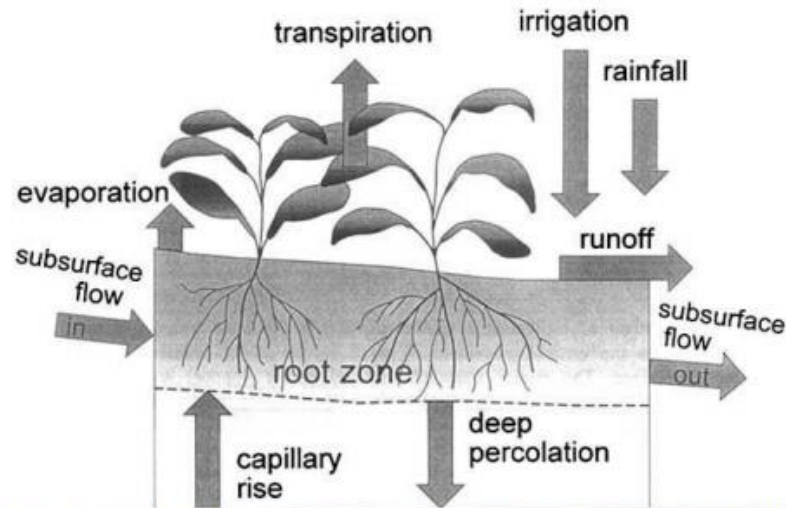


Figure 1: Conceptual full water balance model showing inputs and outputs, Allen et al. (1998)

The full water balance can be described at the soil root zone as:

$$\Delta S = P + I - T - E - RO - DP + CR \pm \Delta SF \quad \text{eq (2)}$$

Where:

ΔS	change in soil water storage [mm]
P	rainfall [mm]
I	irrigation [mm]
T	transpiration [mm]
E	evaporation [mm]
RO	surface runoff [mm]
DP	deep percolation [mm]
CR	capillary rise [mm]
ΔSF	change in subsurface flow [mm]

$$\Delta S = P + I - ET_c \quad \text{eq (3)}$$

Where:

ΔS	change in soil water storage [mm]
P	rainfall [mm]
I	irrigation [mm]
ET_c	crop evapotranspiration [mm]

These simplifications then leave the IrriSAT water balance deficit model as shown in Figure 2. With evapotranspiration (ET_c) removing water from the soil root zone and irrigation and rainfall adding water to the soil root zone.

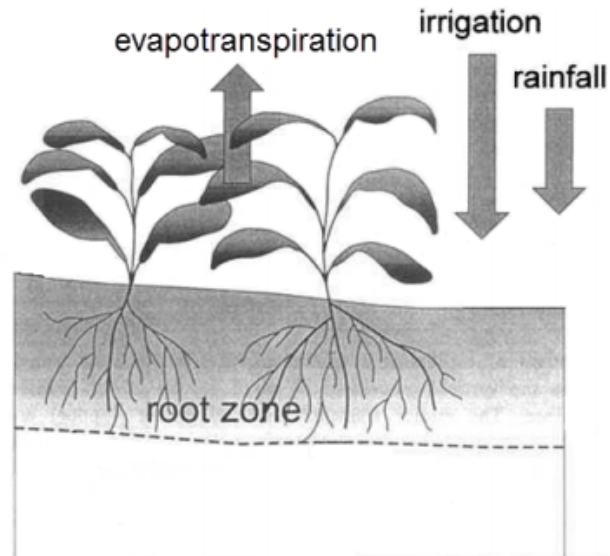


Figure 2: Conceptual IrriSAT water balance model showing inputs and outputs.

When to irrigate? (refill point)

After the readily available water has been used, plant roots cannot extract water as easily from the soil and growth is affected. This point is referred to as the **refill point**. As its name suggests, refill point is the time to irrigate. The drier the soil, the more water it needs to return to field capacity.

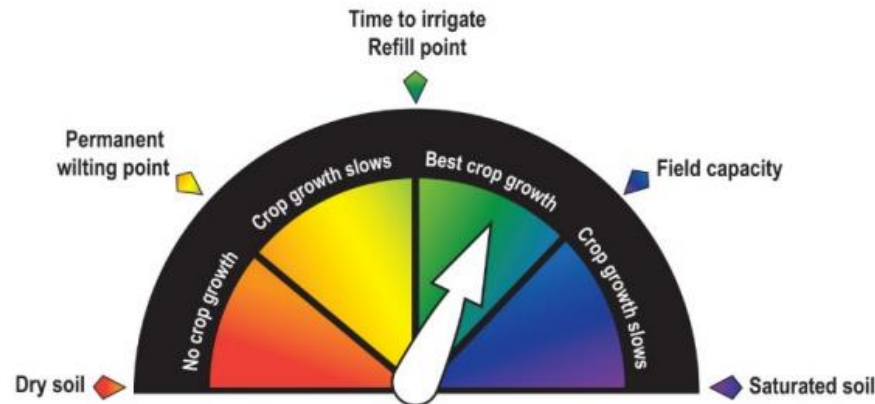
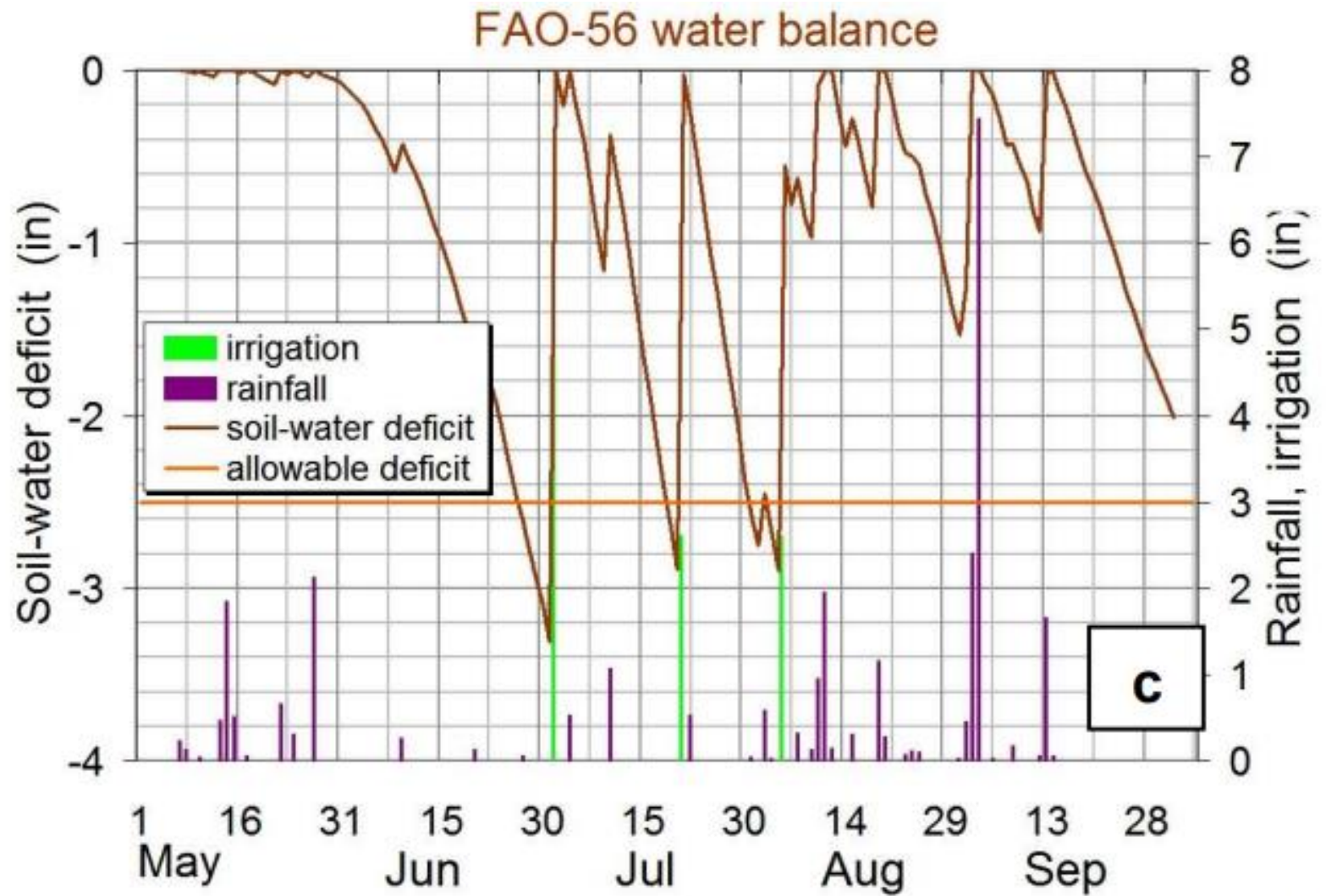


Figure 4: Soil Water 'Fuel Gauge'

Three factors should be taken into account when determining the refill point which are soil type, crop rooting depth and irrigation system. These factors may also change over time, hence the refill point may also change throughout the season.



Crop Water Use (ET_c)

Crop water requirements (ET_c) can be estimated considering a climatic parameter called reference evapotranspiration (ET_o), which represents the evapotranspiration from a standardize vegetated surface, and a crop factor called crop coefficient (K_c) that relates ET_c to ET_o by the equation (Allen et al., 1998):

$$ET_c = ET_o K_c \quad \text{eq (9)}$$

Where:

ET_c	crop evapotranspiration [mm]
ET_o	reference evapotranspiration [mm]
K_c	crop coefficient [-]

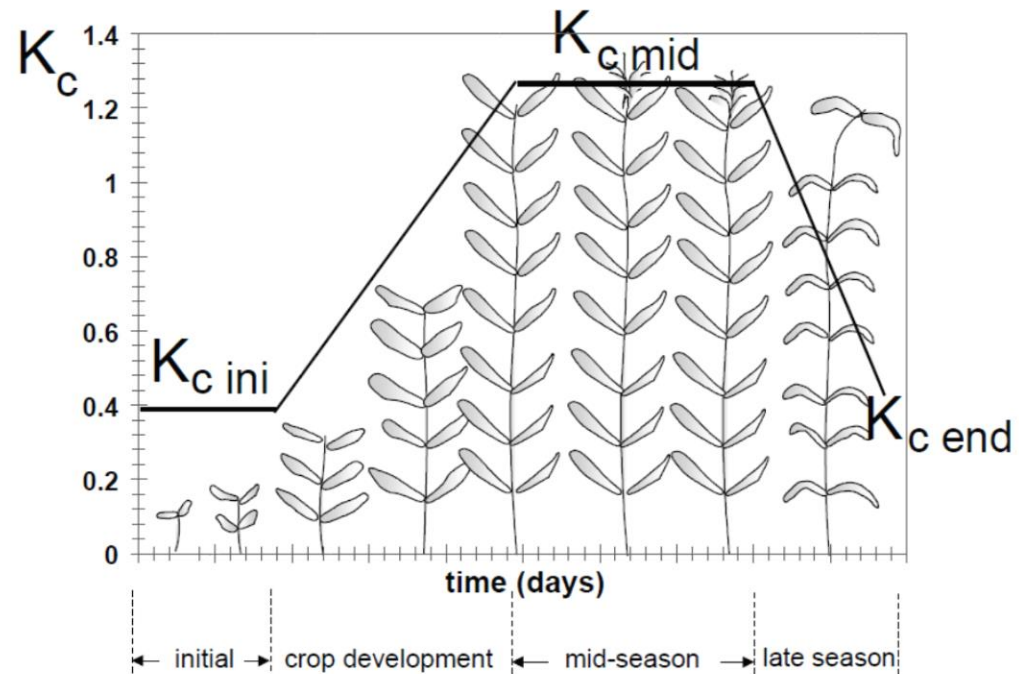
September 2018

IrrisAT Technical Reference

John Hornbuckle¹, Jamie Vleeshouwer², Carlos Ballester¹, Janelle Montgomery³, Robert Hoogers³ & Robert Bridgart²

¹ Deakin University, Centre for Regional and Rural Futures
² CSIRO Land & Water
³ NSW Department of Primary Industries

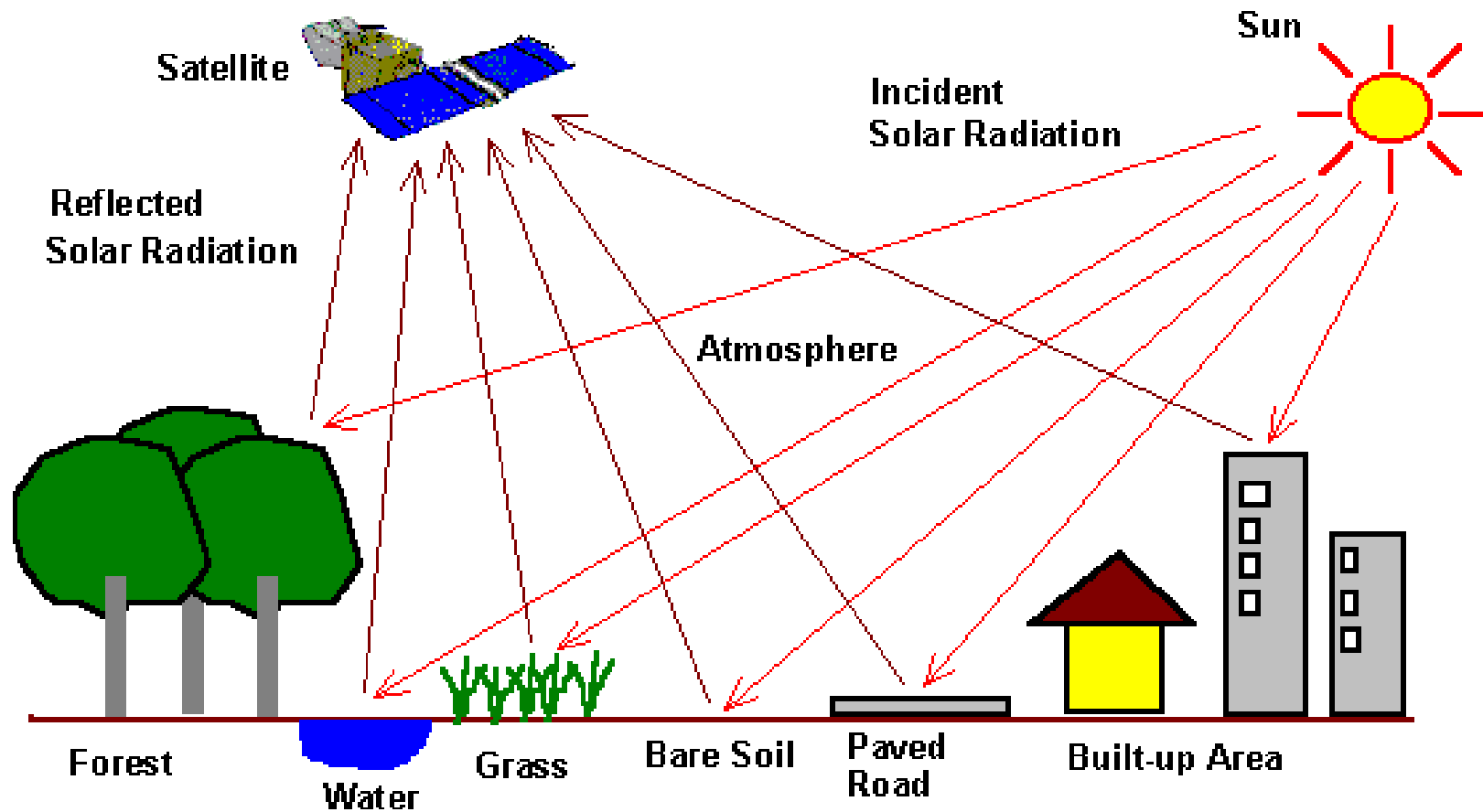
Generalized crop coefficient curve for the single crop coefficient approach



As mentioned above, NDVI has been strongly correlated with crop canopy cover for various crops in semi-arid areas and can be converted to K_c values by the empirical relationships such as (Trout and Johnson 2007):

$$K_c = 1.37 NDVI - 0.086 \quad \text{eq (12)}$$

Remote Sensing



Landsat Missions

- HOME
- LANDSAT MISSIONS**
- Landsat 9
- Landsat 8**
- Landsat 7
- Landsat 6
- Landsat 5
- Landsat 4
- Landsat 3
- Landsat 2
- Landsat 1

Landsat 8

Landsat 8 (formally the Landsat Data Continuity Mission, LDCM) was launched on an Atlas-V rocket from Vandenberg Air Force Base, California on February 11, 2013.

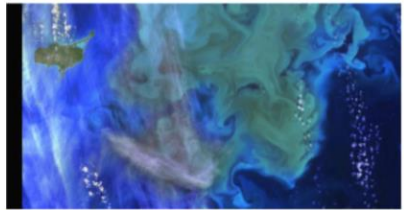
Landsat 8 is the most recently launched Landsat satellite and carries the Operational Land Imager (OLI) and the Thermal Infrared Sensor (TIRS) instruments.

Landsat 8 Launch



[View Launch](#)

Explore More Landsat 8



[Explore USGS](#)



Illustration of the Landsat 8 Satellite.

Landsat 8 orbits the Earth in a sun-synchronous, near-polar orbit, at an altitude of 705.1 (438 mi), inclined at 98.2 degrees, and completes one Earth orbit every 99 minutes. The satellite has a 16-day repeat cycle with an equatorial crossing time: 10:00 a.m. +/- 15 minutes.

Landsat 8 acquires about 740 scenes a day on the [Worldwide Reference System-2](#) (WRS-2) path/row system, with a swath overlap (or sidelap) varying from 7 percent at the equator to maximum of approximately 85 percent at extreme latitudes. A Landsat 8 scene size is 185.1 x 180 km (114 mi x 112 mi).

Data products created from Landsat 8 OLI/TIRS scenes are available to download from [EarthExplorer](#), [GloVis](#), and the [LandLook Viewer](#).

Landsat 8 Instruments

Operational Land Imager (OLI) - Built by Ball Aerospace & Techn

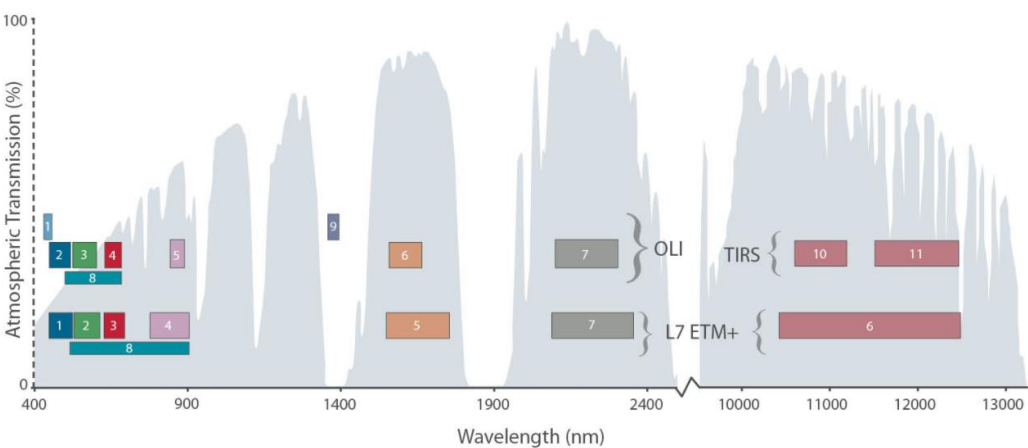
- Nine spectral bands, including a pan band:
 - Band 1 Visible (0.43 - 0.45 μm) 30 m
 - Band 2 Visible (0.450 - 0.51 μm) 30 m
 - Band 3 Visible (0.53 - 0.59 μm) 30 m
 - Band 4 Red (0.64 - 0.67 μm) 30 m
 - Band 5 Near-Infrared (0.85 - 0.88 μm) 30 m
 - Band 6 SWIR 1 (1.57 - 1.65 μm) 30 m
 - Band 7 SWIR 2 (2.11 - 2.29 μm) 30 m
 - Band 8 Panchromatic (PAN) (0.50 - 0.68 μm) 15 m
 - Band 9 Cirrus (1.36 - 1.38 μm) 30 m

OLI captures data with improved radiometric precision over a 12-bit signal to noise ratio. This translates into 4096 potential grey levels, Landsat 1-7 8-bit instruments. Improved signal to noise performance cover state and condition.

The 12-bit data are scaled to 16-bit integers and delivered in the Level 55,000 grey levels, and can be rescaled to the Top of Atmosphere (TOA) radiometric rescaling coefficients provided in the product metadata.

Thermal Infrared Sensor (TIRS) - Built by NASA Goddard Space

- Two spectral bands:
 - Band 10 TIRS 1 (10.6 - 11.19 μm) 100 m
 - Band 11 TIRS 2 (11.5 - 12.51 μm) 100 m



https://earthexplorer.usgs.gov/



EarthExplorer

Help Feedback Login

Search Criteria **Data Sets** Additional Criteria Results

Search Criteria Summary (Show)

Clear Search Criteria

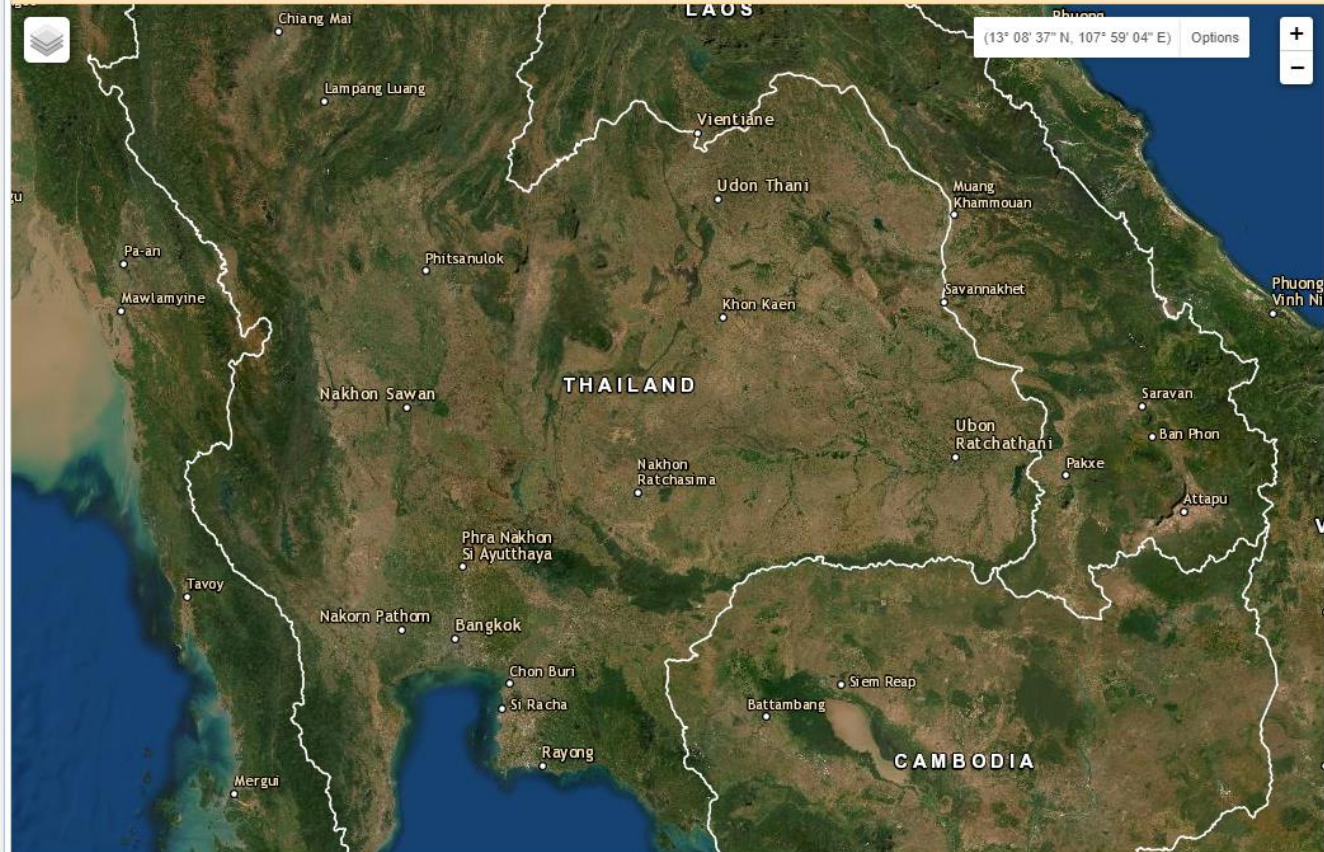
2. Select Your Data Set(s)

Check the boxes for the data set(s) you want to search. When done selecting data set(s), click the *Additional Criteria* or *Results* buttons below. Click the plus sign next to the category name to show a list of data sets.

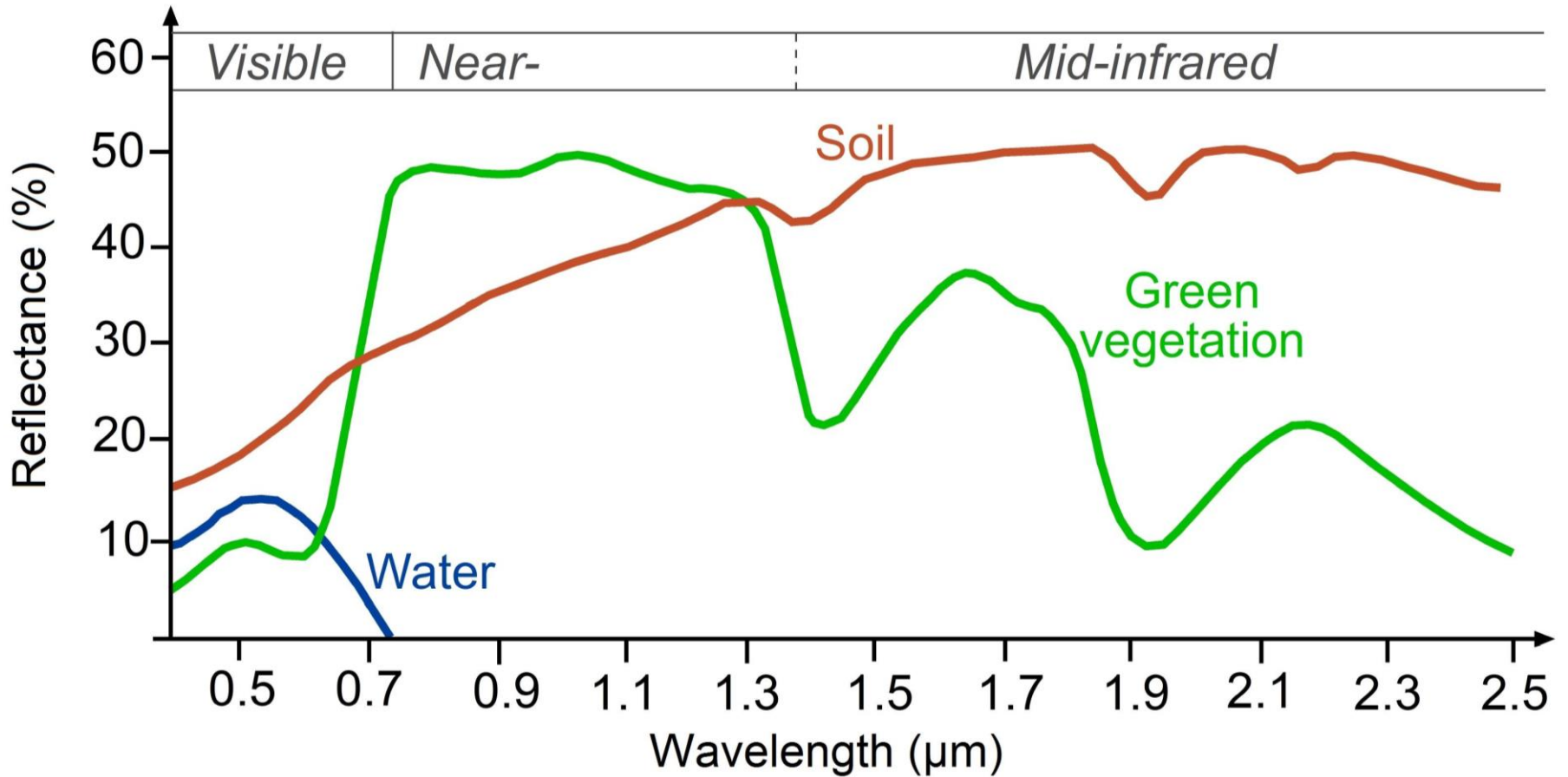
Use Data Set Prefilter (What's This?)

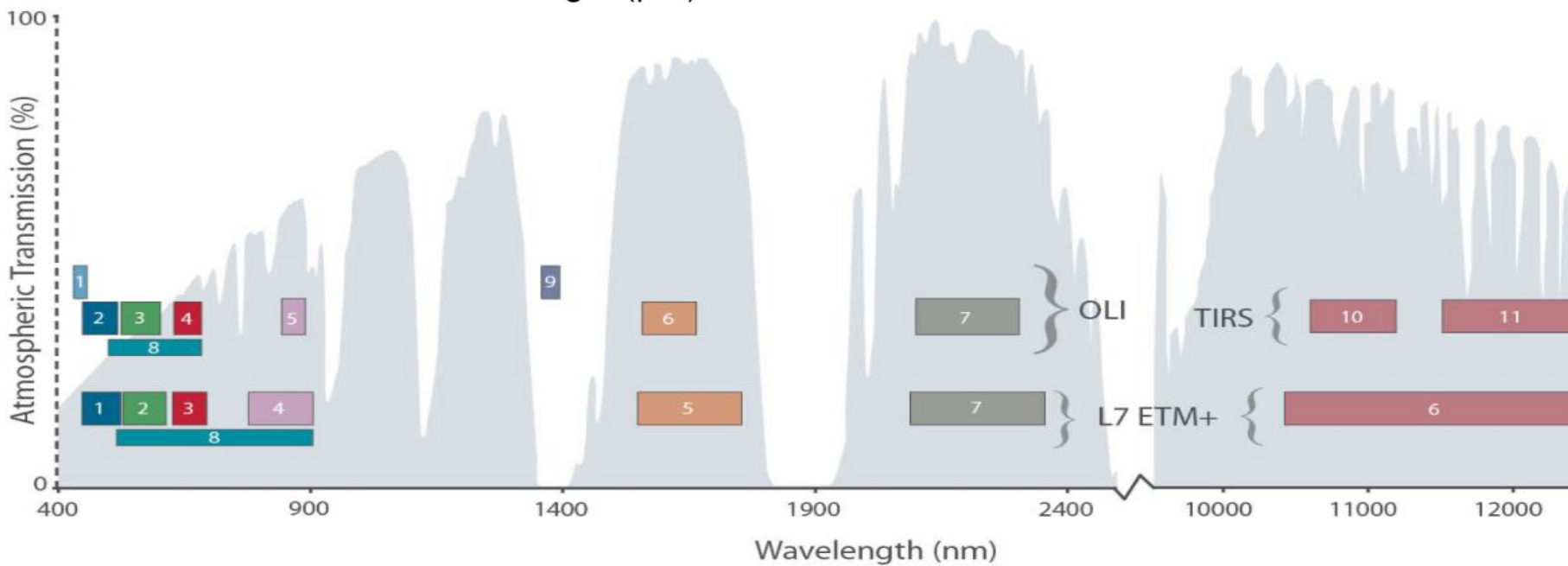
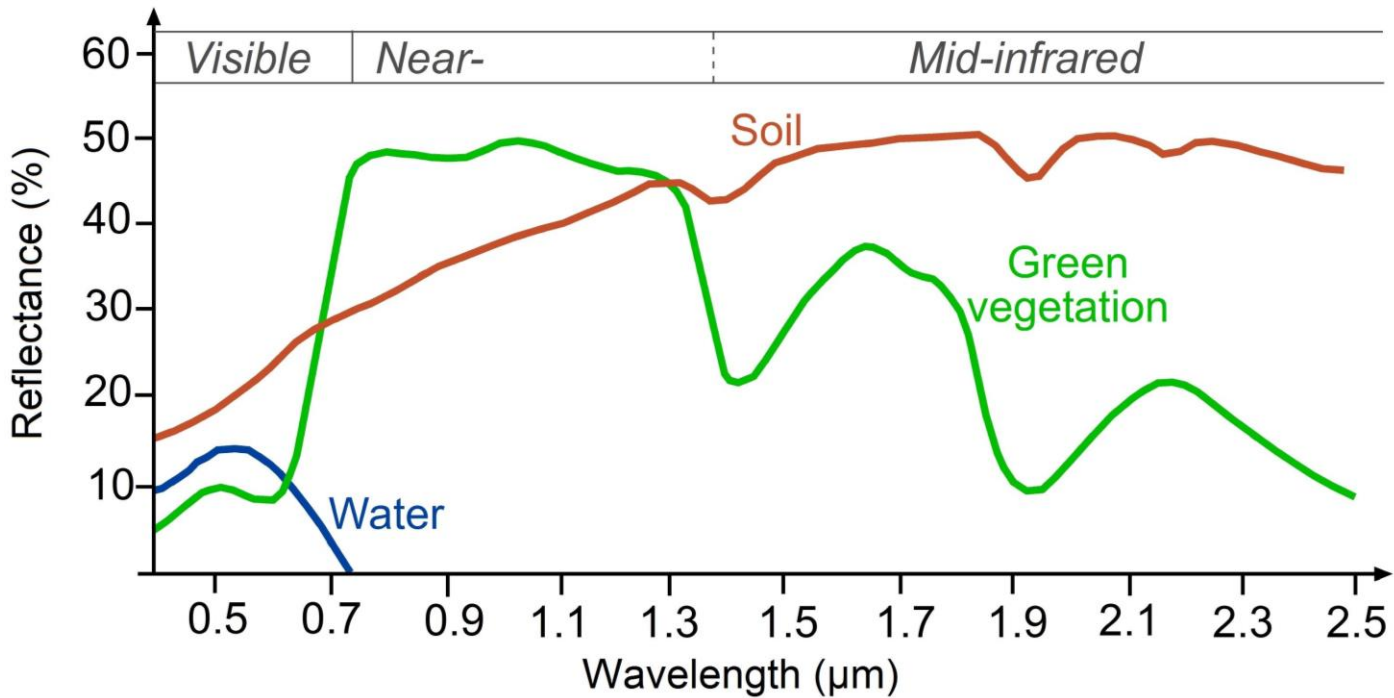
Data Set Search:

- HCMM
- ISERV
- Land Cover
- Landsat
- Landsat Collection 1
 - Landsat Collection 1 Level-3
 - Landsat C1 Analysis Ready Data (ARD)
 - Landsat Collection 1 Level-2 (On-demand)
 - Landsat Collection 1 Level-1
 - Landsat 8 OLI/TIRS C1 Level-1
 - Landsat 7 ETM+ C1 Level-1
 - Landsat 4-5 TM C1 Level-1
 - Landsat 1-5 MSS C1 Level-1
 - Landsat Legacy
- LCMAP
- NASA LPDAAC Collections
- Radar
- Sentinel
- UAS
- Vegetation Monitoring



Spectral Signature





NDVI

Calculation of the Normalized Difference Vegetation Index (NDVI), which is available on-the-fly, comes first. In addition, NDVI is often used around the world to monitor drought, forecast agricultural production, assist in forecasting fire zones and desert offensive maps. Farming apps, like Crop Monitoring, integrate NDVI to facilitate crop scouting and give precision to fertilizer application and irrigation, among other field treatment activities, at specific growth stages. NDVI is preferable for global vegetation monitoring since it helps to compensate for changes in lighting conditions, surface slope, exposure, and other external factors.

NDVI is calculated in accordance with the formula:

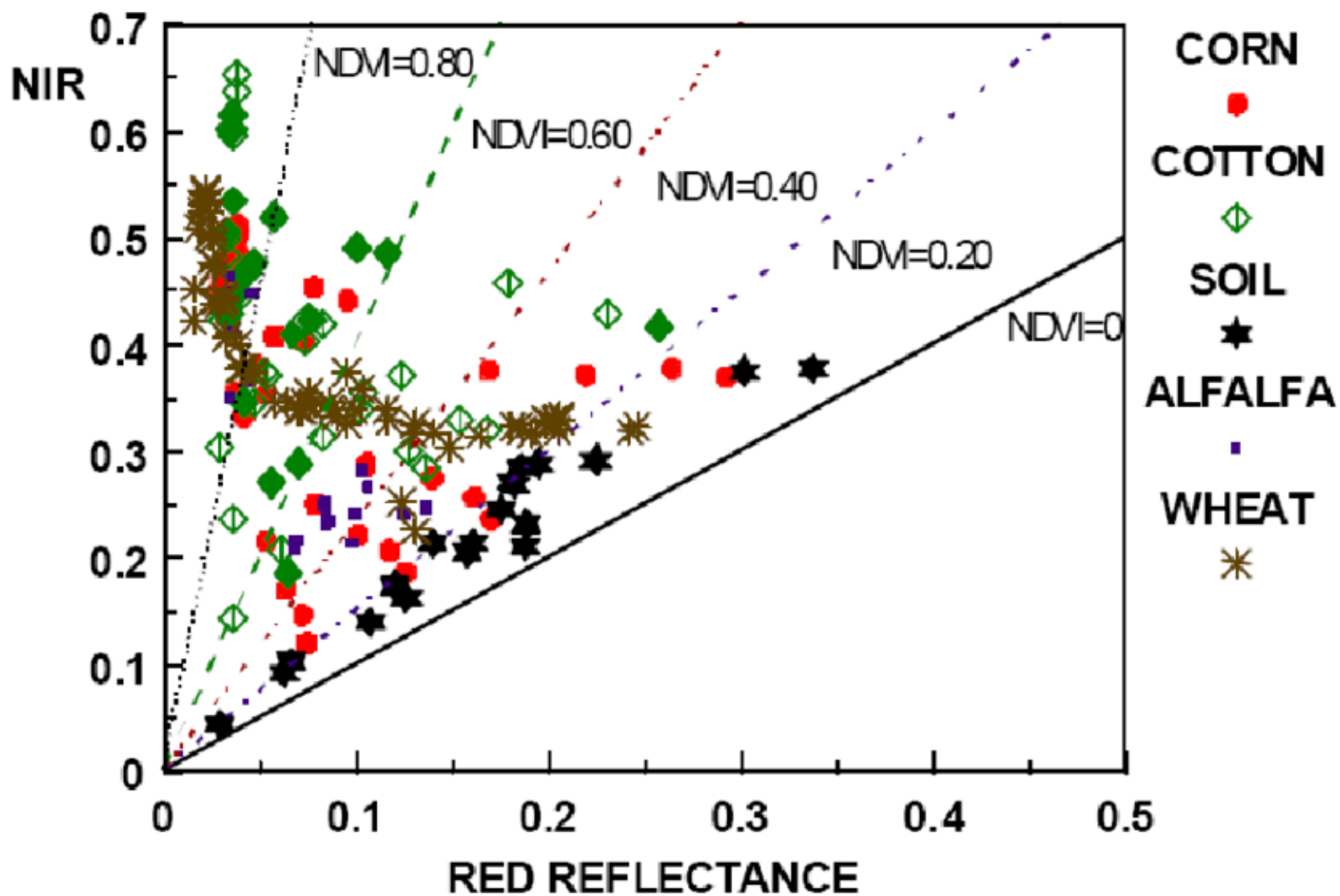
$$NDVI = \frac{NIR - RED}{NIR + RED}$$

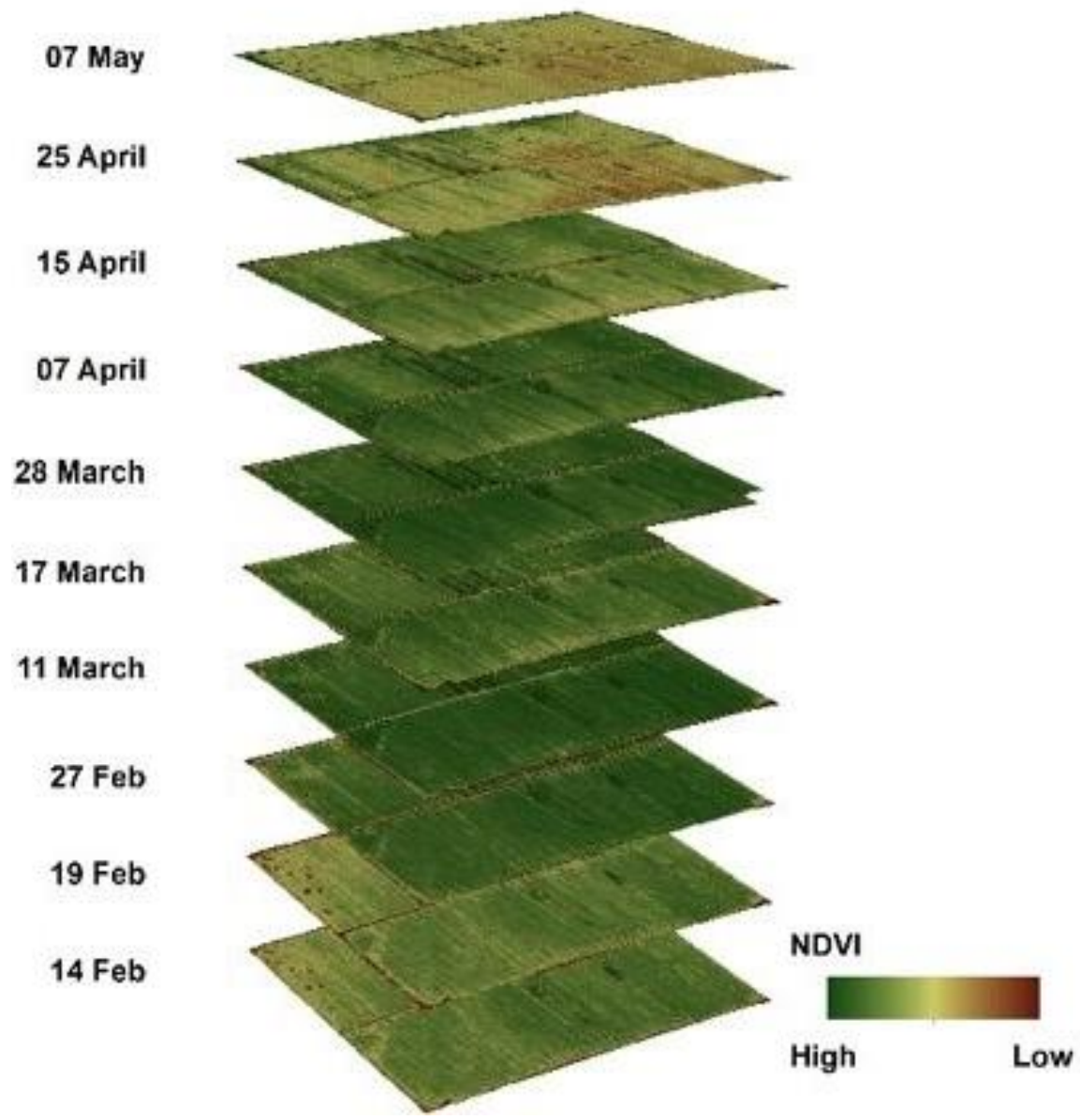
NIR – reflection in the near-infrared spectrum

RED – reflection in the red range of the spectrum

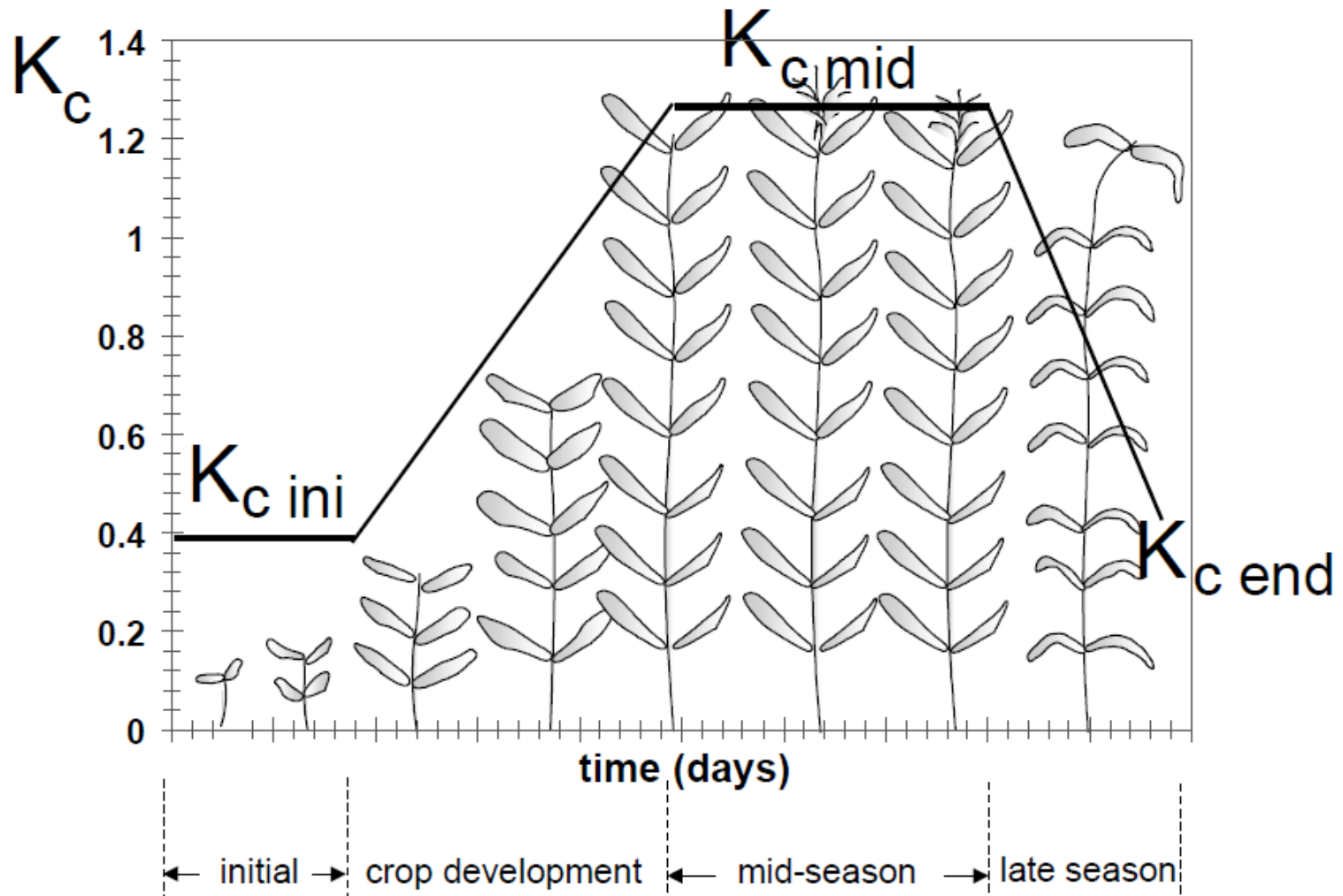
According to this formula, the density of vegetation (NDVI) at a certain point of the image is equal to the difference in the intensities of reflected light in the red and infrared range divided by the sum of these intensities.

This index defines values from -1.0 to 1.0, basically representing greens, where negative values are mainly formed from clouds, water and snow, and values close to zero are primarily formed from rocks and bare soil. Very small values (0.1 or less) of the NDVI function correspond to empty areas of rocks, sand or snow. Moderate values (from 0.2 to 0.3) represent shrubs and meadows, while large values (from 0.6 to 0.8) indicate temperate and tropical forests. Crop Monitoring successfully utilizes this scale to show farmers which parts of their fields have dense, moderate, or sparse vegetation at any given moment.





Generalized crop coefficient curve for the single crop coefficient approach



workshop

- ### Gmail account ###

https://myaccount.google.com/?utm_source=signin_no_continue

- <https://irrisat-cloud.appspot.com/>

https://irrisat-cloud.appspot.com/

The screenshot shows a web browser window with the URL <https://irrisat-cloud.appspot.com/>. The browser's address bar and tabs are visible at the top. The main content area displays a satellite map of a region in Australia, with a date range of "Oct 31 - Nov 7 2020" selected. A sidebar on the left, titled "Maps (Yutthana)", contains a list of map layers with sliders and checkboxes: Landsat 7, Landsat 8, Sentinel 2, Clouds, NDVI, and Kc. The "Kc" layer is checked. A "Satellite" button is located below the list. On the right side of the map, there is a legend for the "Kc" layer, showing a color scale from 0 (brown) to 1.2 (blue), with a midpoint at 0.6. The map itself shows a network of roads and agricultural fields, with several blue location pins. The bottom of the browser window shows a Windows taskbar with various application icons and the system tray displaying the time as 11:33 PM on 11/10/2020.

การสร้างแบบจำลองสำหรับบริหารจัดการเชื่อมด้วยโปรแกรม MATLAB

อาจารย์ ดร.ยุธนา พันธุ์กมลศิลป์

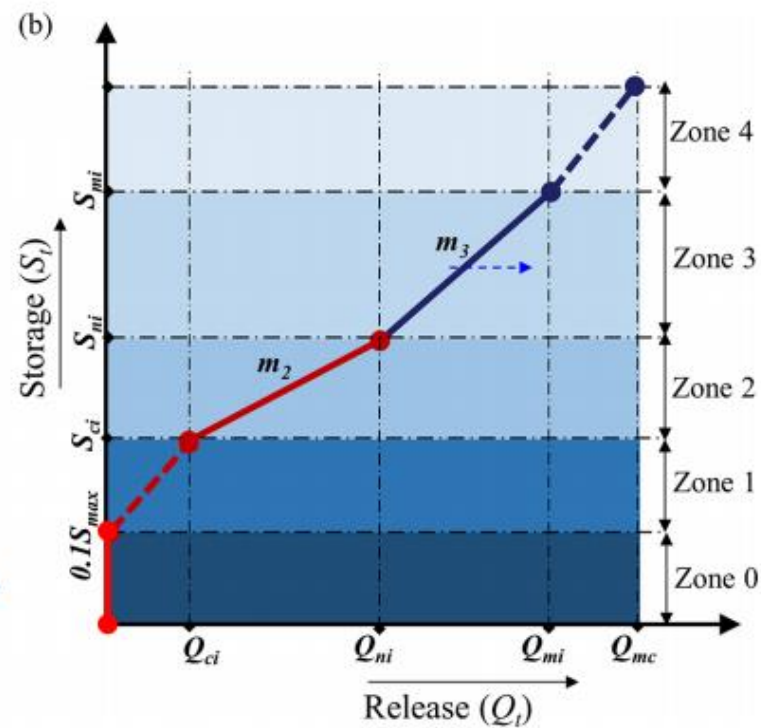
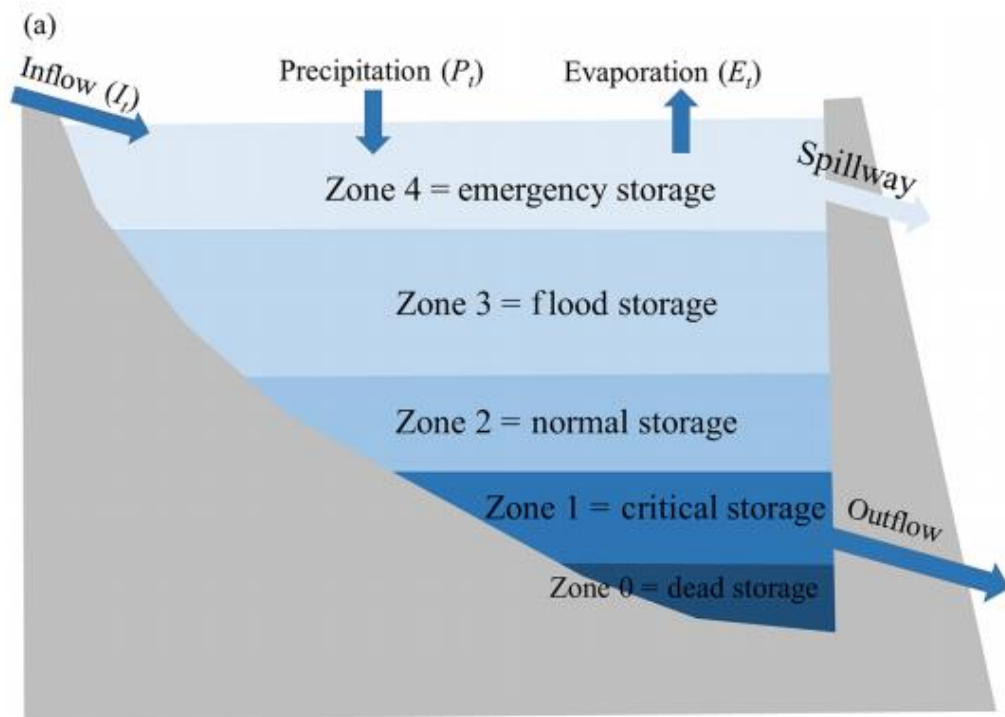
สาขาวิชาวิศวกรรมสิ่งแวดล้อมและการจัดการภัยพิบัติ

มหาวิทยาลัยมหิดล วิทยาเขตกาญจนบุรี

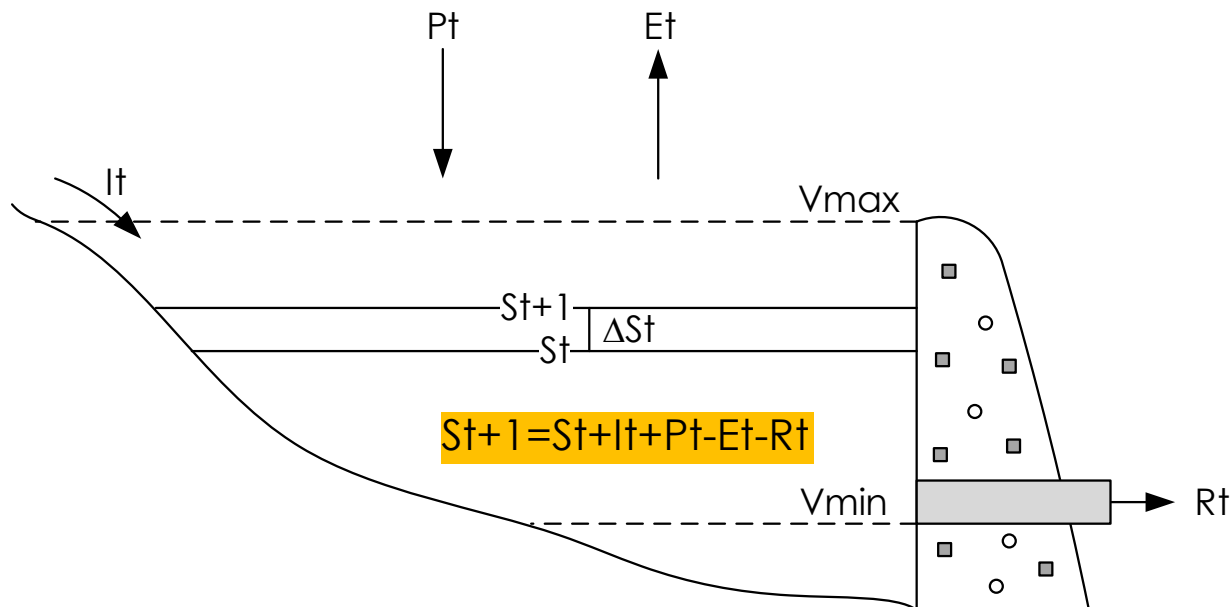
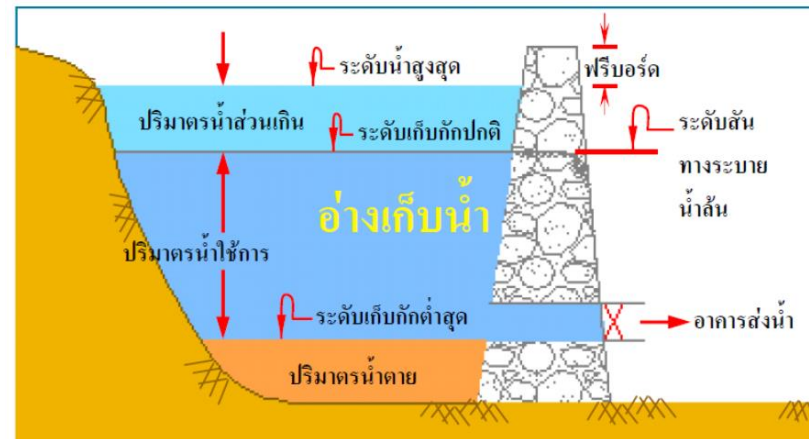
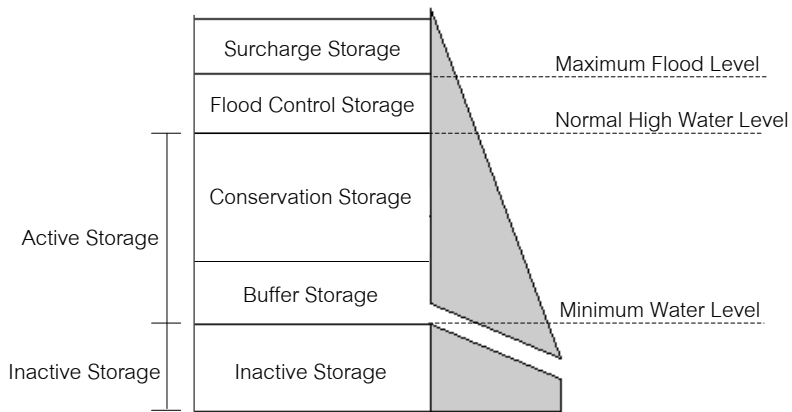
12 พฤศจิกายน 2563

Bhumibol Dam
The largest concrete arch
dam in Thailand

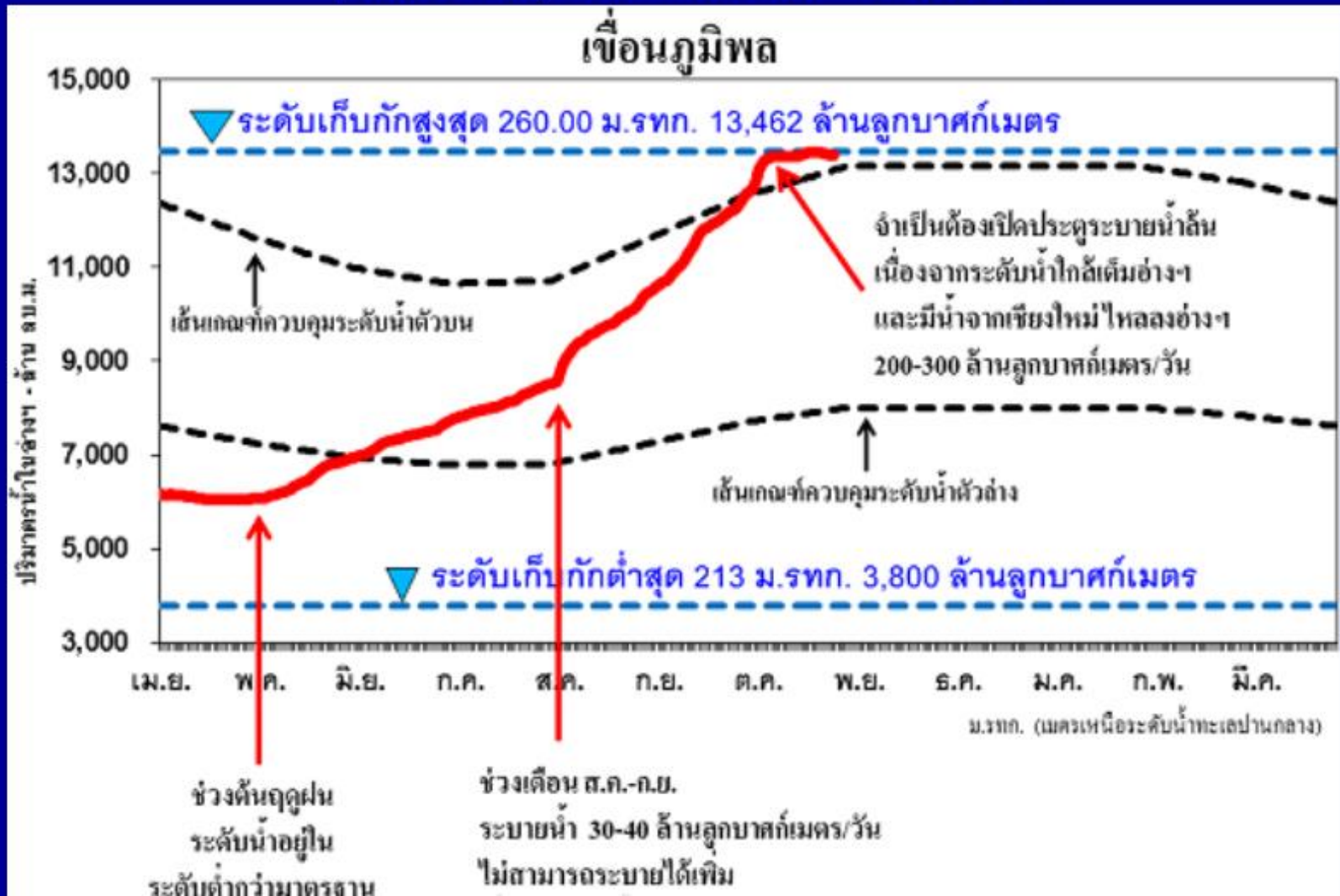




Reservoir Design/Reservoir Sizing



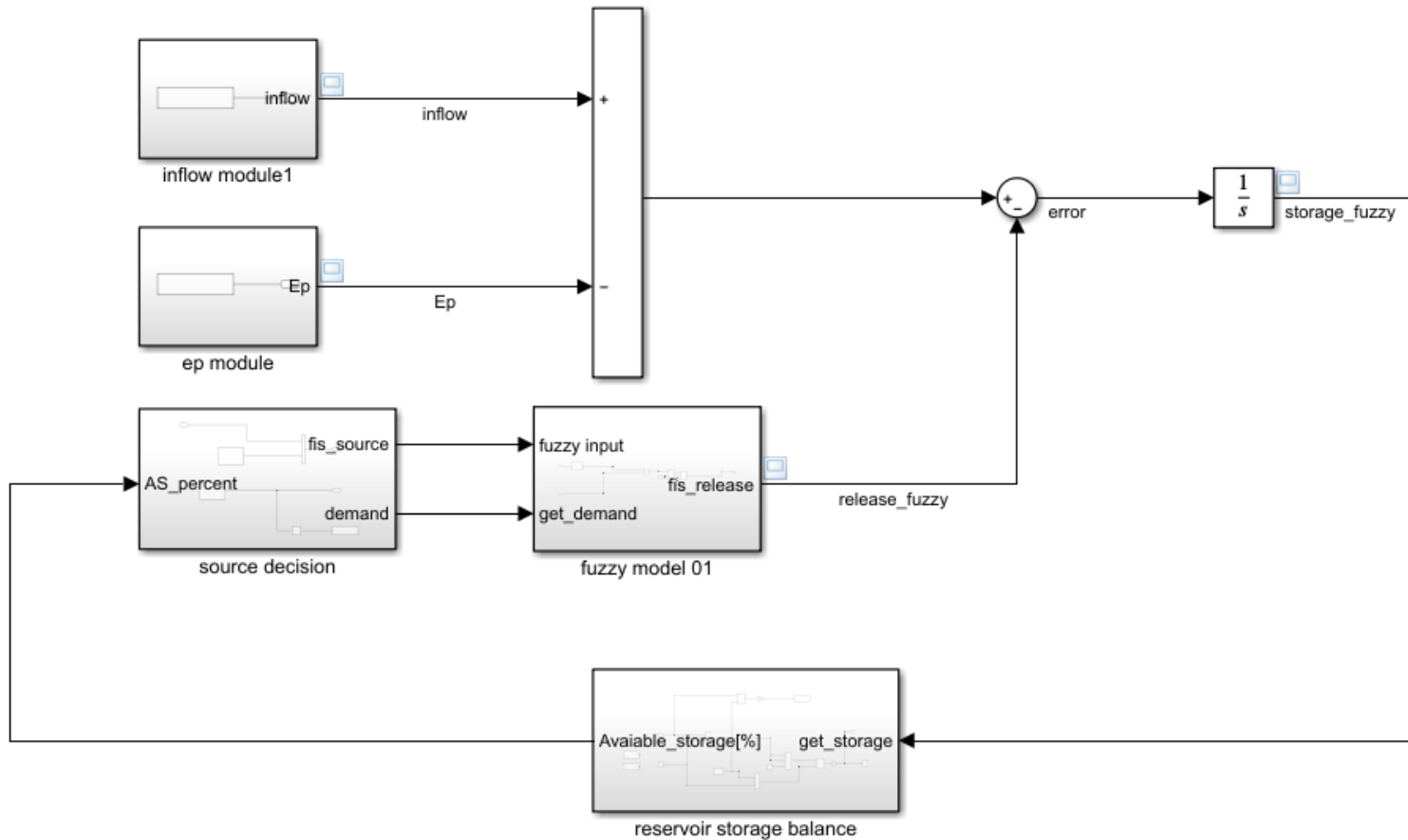
การบริหารจัดการน้ำเขื่อนภูมิพลโดยเกณฑ์ระดับน้ำควบคุม (Rule Curve) ในปี พ.ศ. 2554 ซึ่งเป็นปีน้ำมาก



Practice : fuzzy_release_easygo_01

- MATLAB feature
- Simulink library
- Fuzzy logic control

fuzzy_release_easygo_01



Fuzzy Logic Designer: release_based_demand

File Edit View

FIS Name: release based dema FIS Type: mamdani

And method: min Or method: max Implication: min Aggregation: max Defuzzification: centroid

Current Variable: Name: Inflow Type: input Range: [0 300]

Help Close

Updating Membership Function Editor

Membership Function Editor: release_based_demand

File Edit View

FIS Variables: available_storage, Release, Inflow

Membership function plots plot points: 181

Current Variable: Name: available_storage Type: input Range: [0 100] Display Range: [0 100]

Current Membership Function (click on MF to select): Name: Low Type: trapmf Params: [-10 -7.54 39 52.5]

Help Close

Selected variable "available_storage"

Membership Function Editor: release_based_demand

File Edit View

FIS Variables: available_storage, Release, Inflow

Membership function plots plot points: 181

Current Variable: Name: Inflow Type: input Range: [0 300] Display Range: [0 300]

Current Membership Function (click on MF to select): Name: Low Type: trapmf Params: [-150 -30 30 90]

Help Close

Selected variable "Inflow"

Membership Function Editor: release_based_demand

File Edit View

FIS Variables: available_storage, Release, Inflow

Membership function plots plot points: 181

Current Variable: Name: Release Type: output Range: [0.8 1.2] Display Range: [0.8 1.2]

Current Membership Function (click on MF to select): Name: less Type: trapmf Params: [0.76 0.8 0.9 0.9377]

Help Close

Selected variable "Release"

Practice : anfis_release_easygo_01

- MATLAB feature
- Simulink library
- ANFIS

Fuzzy Logic Designer: fis_anfis_01

File Edit View

input1
input2
input3

fis_anfis_01
(sugeno)

f(u)
output

FIS Name: fis_anfis_01 FIS Type: sugeno

And method	prod	Current Variable	
Or method	probor	Name	
Implication	min	Type	
Aggregation	max	Range	
Defuzzification	wtaver		

Help Close

Saved FIS "fis_anfis_01" to file

```
3. If (input1 is in1mf1) and (input2 is in2mf1) and (input3 is in3mf3) then (output is out1mf3) (1)
4. If (input1 is in1mf1) and (input2 is in2mf2) and (input3 is in3mf1) then (output is out1mf4) (1)
5. If (input1 is in1mf1) and (input2 is in2mf2) and (input3 is in3mf2) then (output is out1mf5) (1)
6. If (input1 is in1mf1) and (input2 is in2mf2) and (input3 is in3mf3) then (output is out1mf6) (1)
7. If (input1 is in1mf1) and (input2 is in2mf3) and (input3 is in3mf1) then (output is out1mf7) (1)
8. If (input1 is in1mf1) and (input2 is in2mf3) and (input3 is in3mf2) then (output is out1mf8) (1)
9. If (input1 is in1mf1) and (input2 is in2mf3) and (input3 is in3mf3) then (output is out1mf9) (1)
10. If (input1 is in1mf2) and (input2 is in2mf1) and (input3 is in3mf1) then (output is out1mf10) (1)
11. If (input1 is in1mf2) and (input2 is in2mf1) and (input3 is in3mf2) then (output is out1mf11) (1)
12. If (input1 is in1mf2) and (input2 is in2mf1) and (input3 is in3mf3) then (output is out1mf12) (1)
13. If (input1 is in1mf2) and (input2 is in2mf2) and (input3 is in3mf1) then (output is out1mf13) (1)
14. If (input1 is in1mf2) and (input2 is in2mf2) and (input3 is in3mf2) then (output is out1mf14) (1)
15. If (input1 is in1mf2) and (input2 is in2mf2) and (input3 is in3mf3) then (output is out1mf15) (1)
16. If (input1 is in1mf2) and (input2 is in2mf3) and (input3 is in3mf1) then (output is out1mf16) (1)
17. If (input1 is in1mf2) and (input2 is in2mf3) and (input3 is in3mf2) then (output is out1mf17) (1)
18. If (input1 is in1mf2) and (input2 is in2mf3) and (input3 is in3mf3) then (output is out1mf18) (1)
19. If (input1 is in1mf3) and (input2 is in2mf1) and (input3 is in3mf1) then (output is out1mf19) (1)
20. If (input1 is in1mf3) and (input2 is in2mf1) and (input3 is in3mf2) then (output is out1mf20) (1)
21. If (input1 is in1mf3) and (input2 is in2mf1) and (input3 is in3mf3) then (output is out1mf21) (1)
22. If (input1 is in1mf3) and (input2 is in2mf2) and (input3 is in3mf1) then (output is out1mf22) (1)
23. If (input1 is in1mf3) and (input2 is in2mf2) and (input3 is in3mf2) then (output is out1mf23) (1)
24. If (input1 is in1mf3) and (input2 is in2mf2) and (input3 is in3mf3) then (output is out1mf24) (1)
25. If (input1 is in1mf3) and (input2 is in2mf3) and (input3 is in3mf1) then (output is out1mf25) (1)
26. If (input1 is in1mf3) and (input2 is in2mf3) and (input3 is in3mf2) then (output is out1mf26) (1)
27. If (input1 is in1mf3) and (input2 is in2mf3) and (input3 is in3mf3) then (output is out1mf27) (1)
```

if	and	and	Then
input1 is	input2 is	input3 is	output is
<input type="text" value="in1mf1"/> in1mf2 in1mf3 none	<input type="text" value="in2mf1"/> in2mf2 in2mf3 none	<input type="text" value="in3mf1"/> in3mf2 in3mf3 none	<input type="text" value="out1mf1"/> out1mf2 out1mf3 out1mf4 out1mf5 out1mf6 out1mf7 out1mf8 out1mf9 out1mf10 out1mf11 out1mf12 out1mf13 out1mf14

<input type="checkbox"/> not	<input type="checkbox"/> not	<input type="checkbox"/> not	<input type="checkbox"/> not
------------------------------	------------------------------	------------------------------	------------------------------

Connection	Weight:					
<input type="radio"/> or						
<input checked="" type="radio"/> and	1	Delete rule	Add rule	Change rule	<<	>>

DEBUG MODELING FORMAT APPS

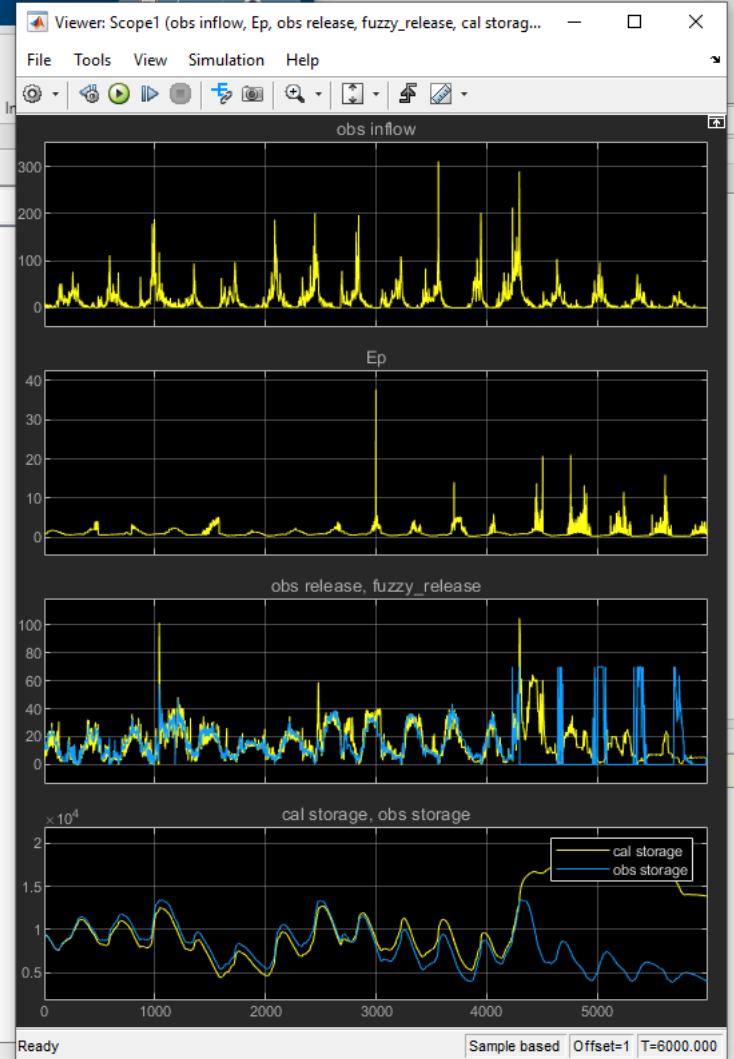
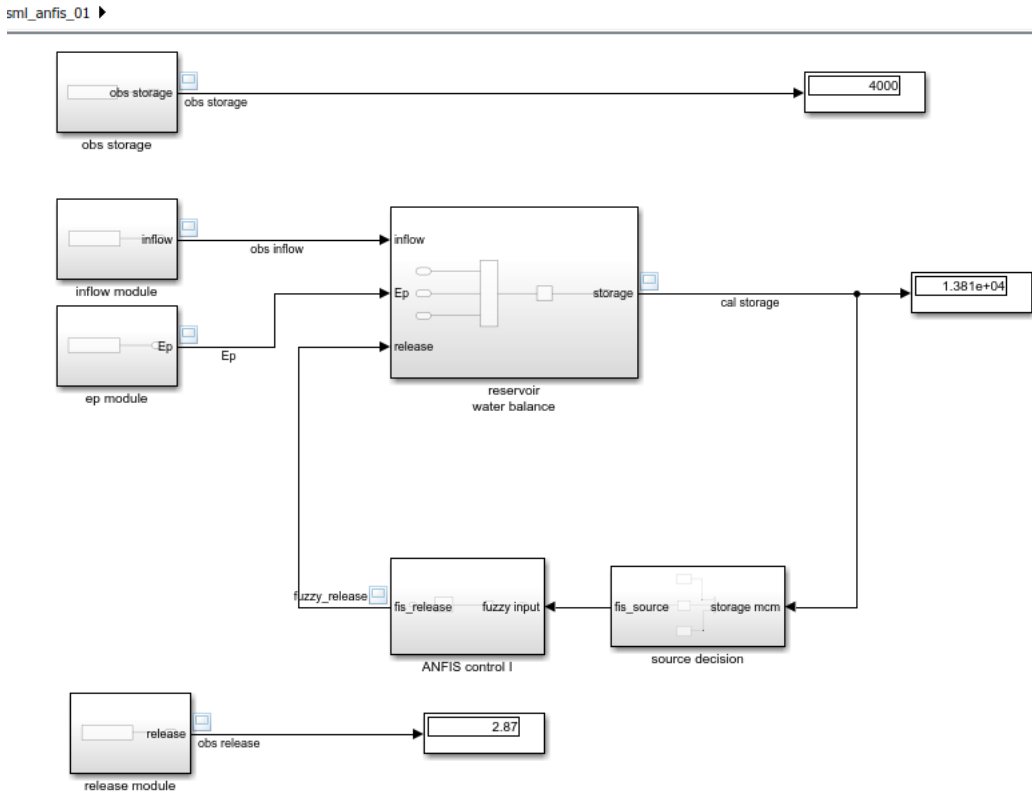
Library Browser Log Signals Add Viewer Signal Table

LIBRARY PREPARE

Stop Time: 6000 Normal Step Back Run Step Forward Stop

Fast Restart SIMULATE

sml_anfis_01





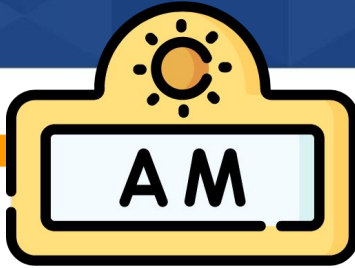
Mahidol University
Wisdom of the Land



“การบริหารจัดการเชื่อมด้วยเทคนิคปัญญาประดิษฐ์
เบื้องต้นด้วยภาษา R และ MiniZinc”

การฝึกอบรมเชิงปฏิบัติการ (Day 2)

Instructor: Jidapa Krajangka
Email: jidapa.kra@mahidol.edu



*Dr. Jidapa
(Pa)
9:30 - 12:00*

Basic Machine
Learning Concept

+

Building a
model with R



*Dr. Wudhichart
(Wud)
13:00 - 16:00*

Optimization

+

Building a
model with MiniZinc



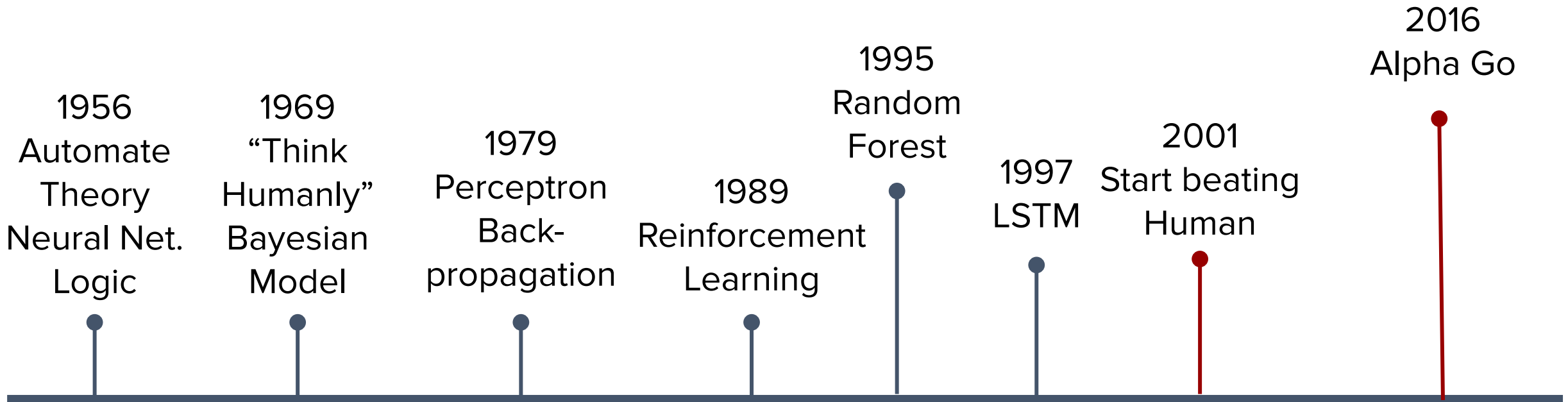
- Introduction + Warm Up
- Hydroinformatics + Machine Learning
- Case Study: ML in Reservoir Management
- Intro to R + Kaggle Notebook
- Demo and Workshop



- **Artificial intelligence (AI)** refers to the simulation of human intelligence in machines that are **programmed to think** like humans and mimic their actions.
- Four different approaches historically defined the field of AI:
 - **Thinking humanly**
 - **Thinking rationally**
 - **Acting humanly**
 - **Acting rationally**



(History of Artificial Intelligence)



Ref: <https://cloud.withgoogle.com/build/data-analytics/explore-history-machine-learning/>

Ref: https://en.wikipedia.org/wiki/Timeline_of_machine_learning



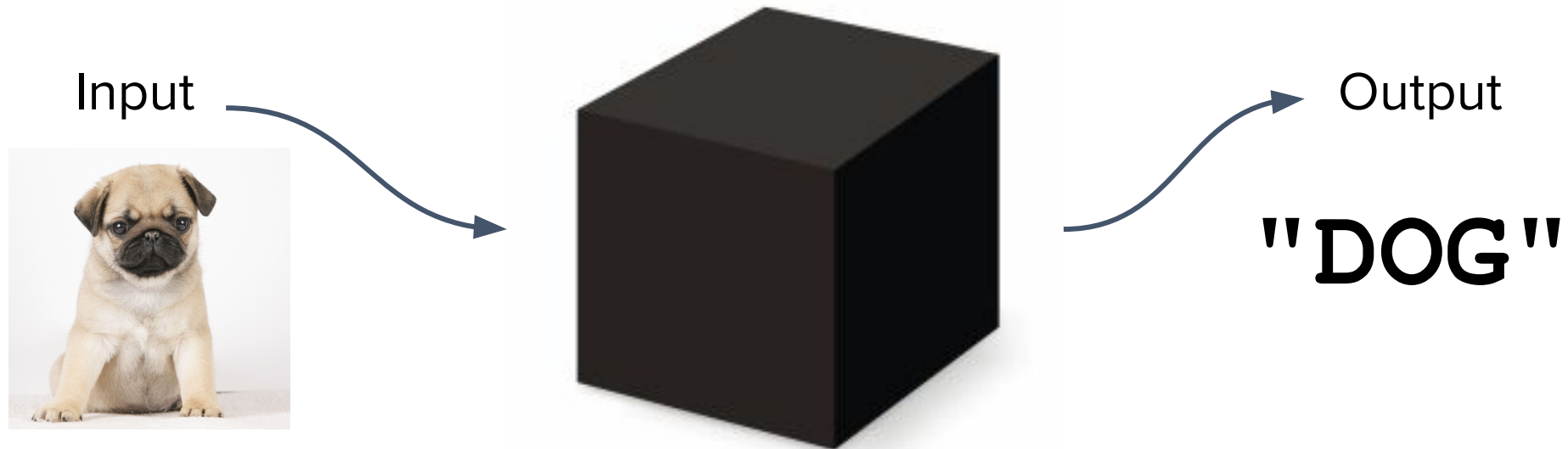
“A field of study that gives computers the ability to learn without being explicitly program” -- Arthur Samuel (1959)

“Machine learning is a subset of AI. That is, all machine learning counts as AI, but not all AI counts as machine learning. For example, symbolic logic – rules engines, expert systems and knowledge graphs – could all be described as AI, and none of them are machine learning.”

Ref: <https://pathmind.com/wiki/ai-vs-machine-learning-vs-deep-learning>

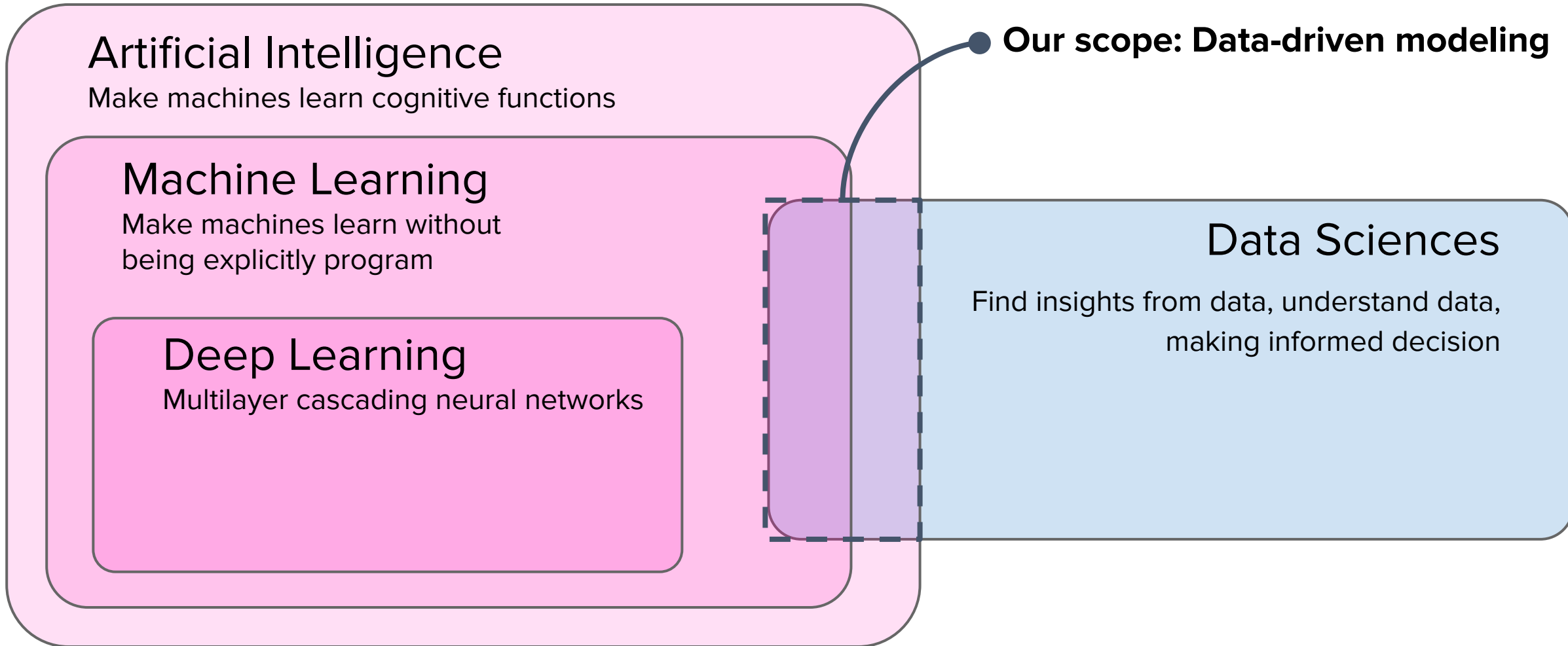


Machine learning is frequently referred to as a **black box**—data goes in, output come out, but the processes between input and output are opaque



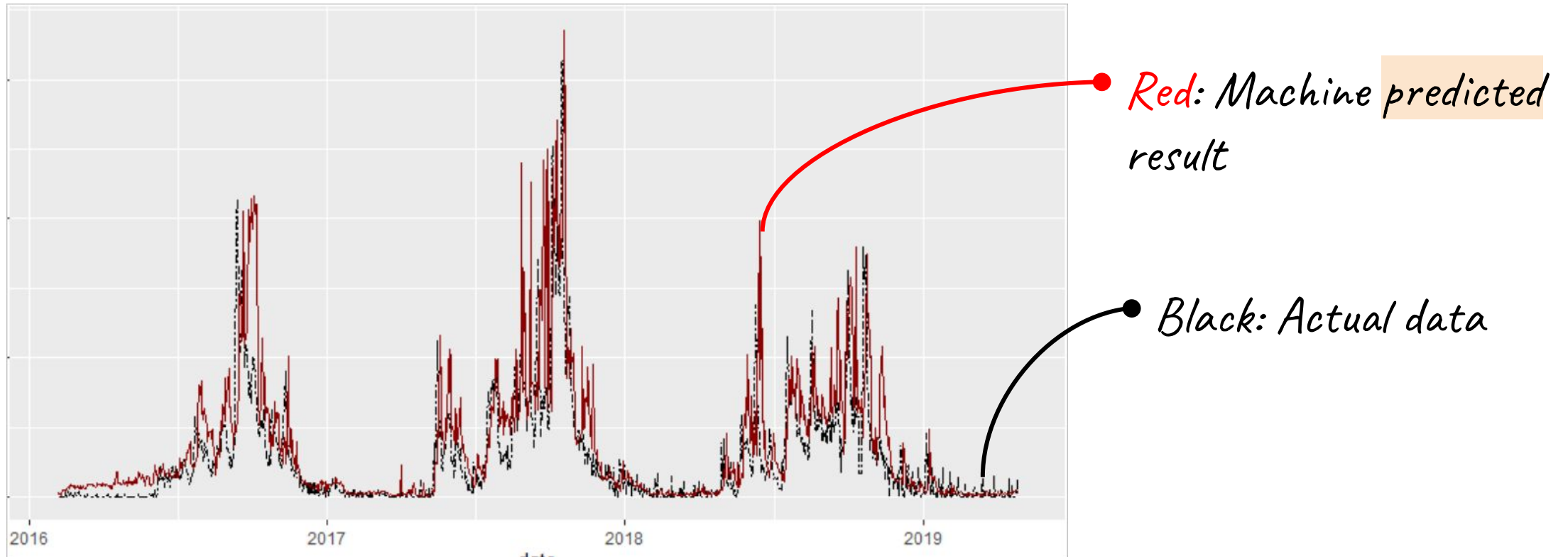


- **Hydrological models** can be characterised as *physical*, *mathematical* (including lumped conceptual and distributed physically based models) and *empirical* (analysis of time series)
- **Data-driven modeling** is the new approach in particular finding **connections between the variables** (input, internal and output variables) without explicit knowledge of the physical behaviour (aka. **Machine Learning**)





- Train the machine to map the given input data to the desired output (**supervised learning**)



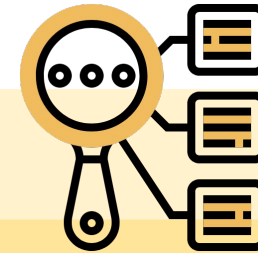


Two main tasks in supervised learning



Regression

- Predict a **continuous** variable
- Input data can be both categorical or continuous variables
- **Synonym: Prediction, Forecasting**



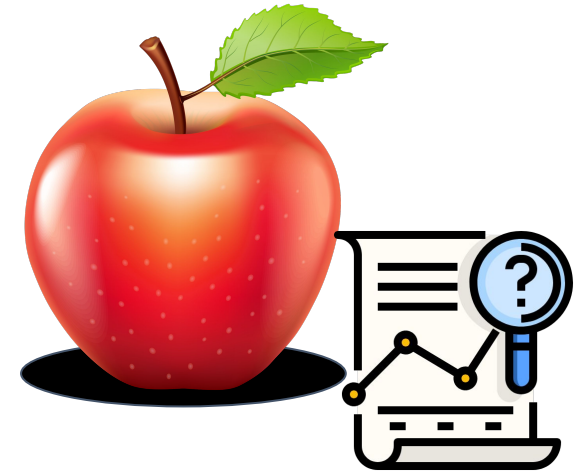
Classification

- Predict a **class/categorical** variable
- Input data can be both categorical or continuous variables



- A **regression** problem requires the *prediction* of a quantity (numerical)

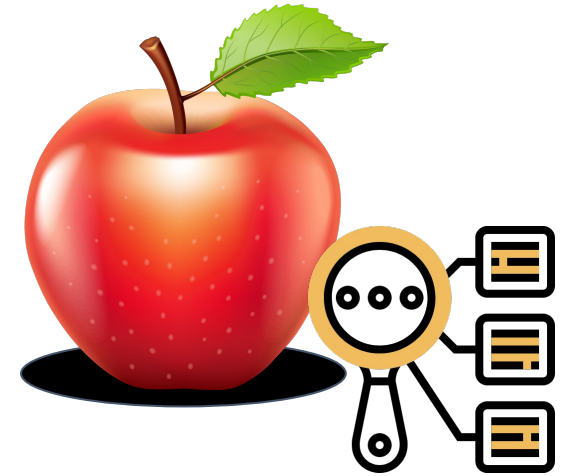
Given an apple height 3" and weight 200 grams, the amount of the carb should be ~ 12.5



- A regression can have real-valued or discrete or mixed input predictors.



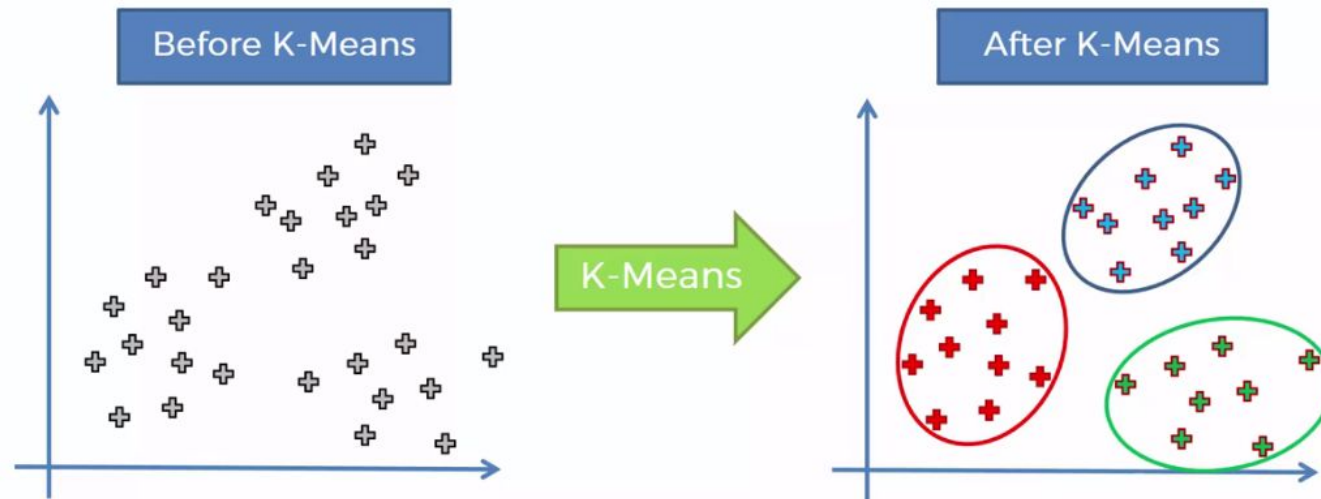
- A **classification** problem requires that examples be classified into classes
 - **Binary** "Apple" or "Not-apple"
 - **Multi-class** "Gala" or "Fuji" or "Winesap"
 - **Multi-label** "Apple" and "Fruit" and "Red"
- A classification can have real-valued or discrete or mixed input predictors.





Let the machine discover the pattern/underlying structure in the data given the inputs

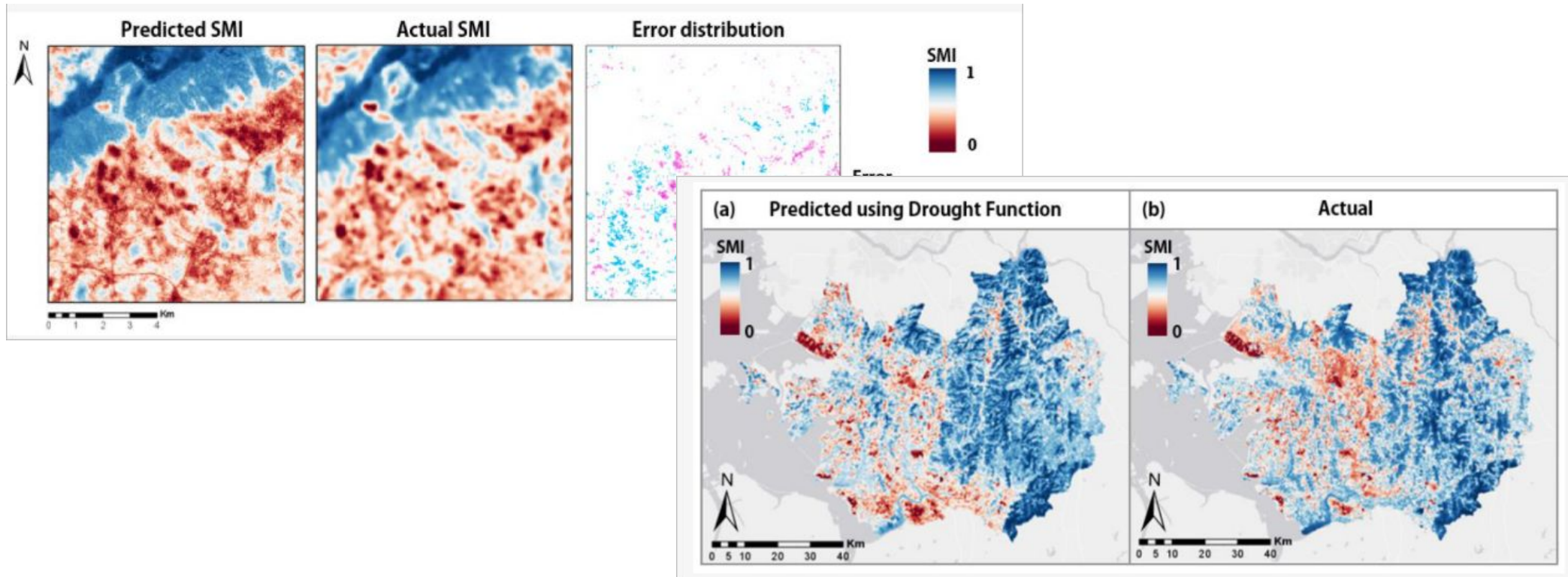
Task: Clustering





“Prediction of Severe Drought Area Based on Random Forest: Using Satellite Image and Topography Data”

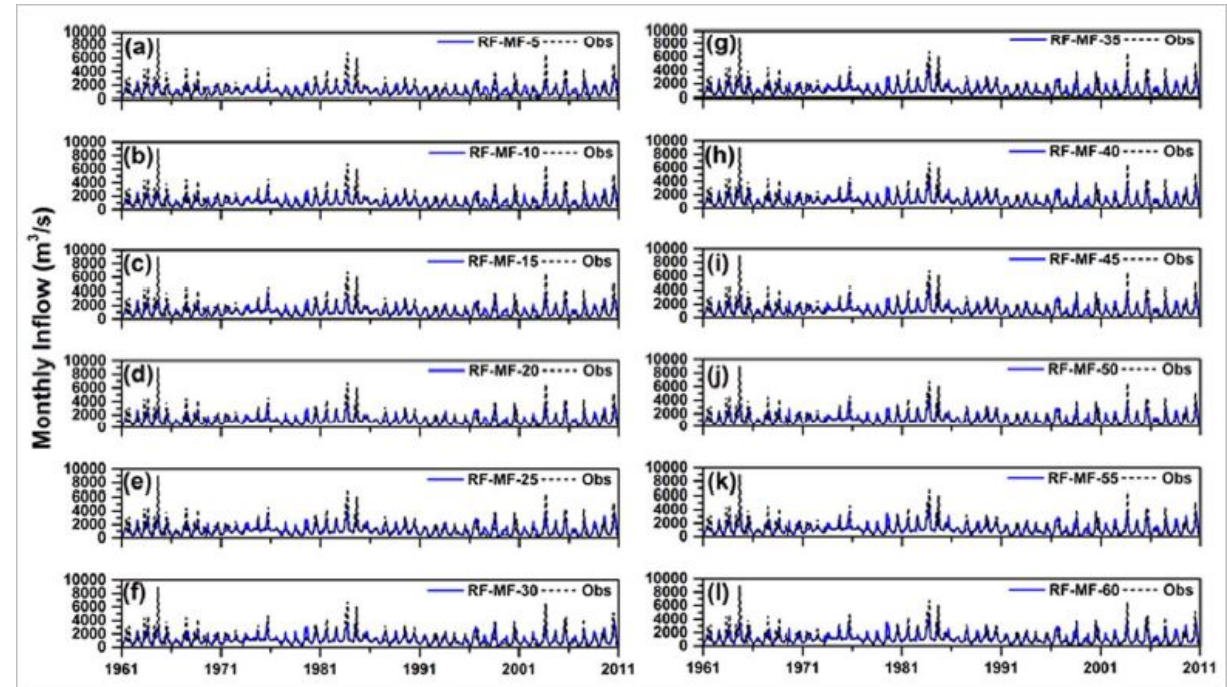
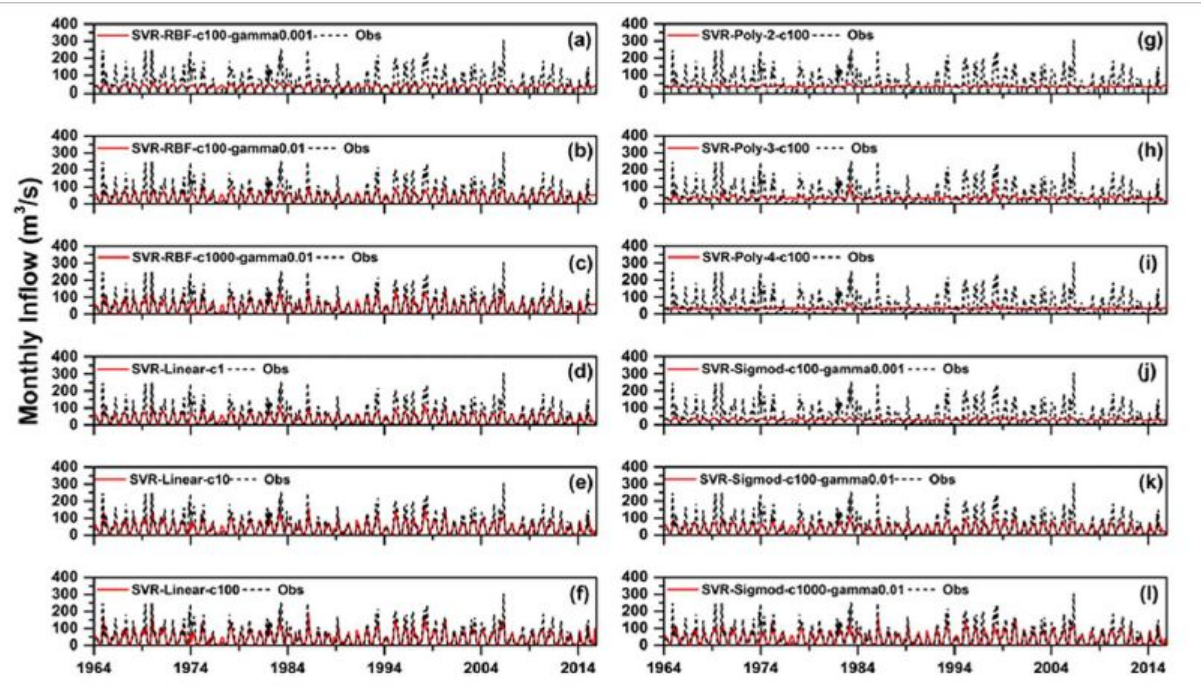
Ref: <https://doi.org/10.3390/w11040705>

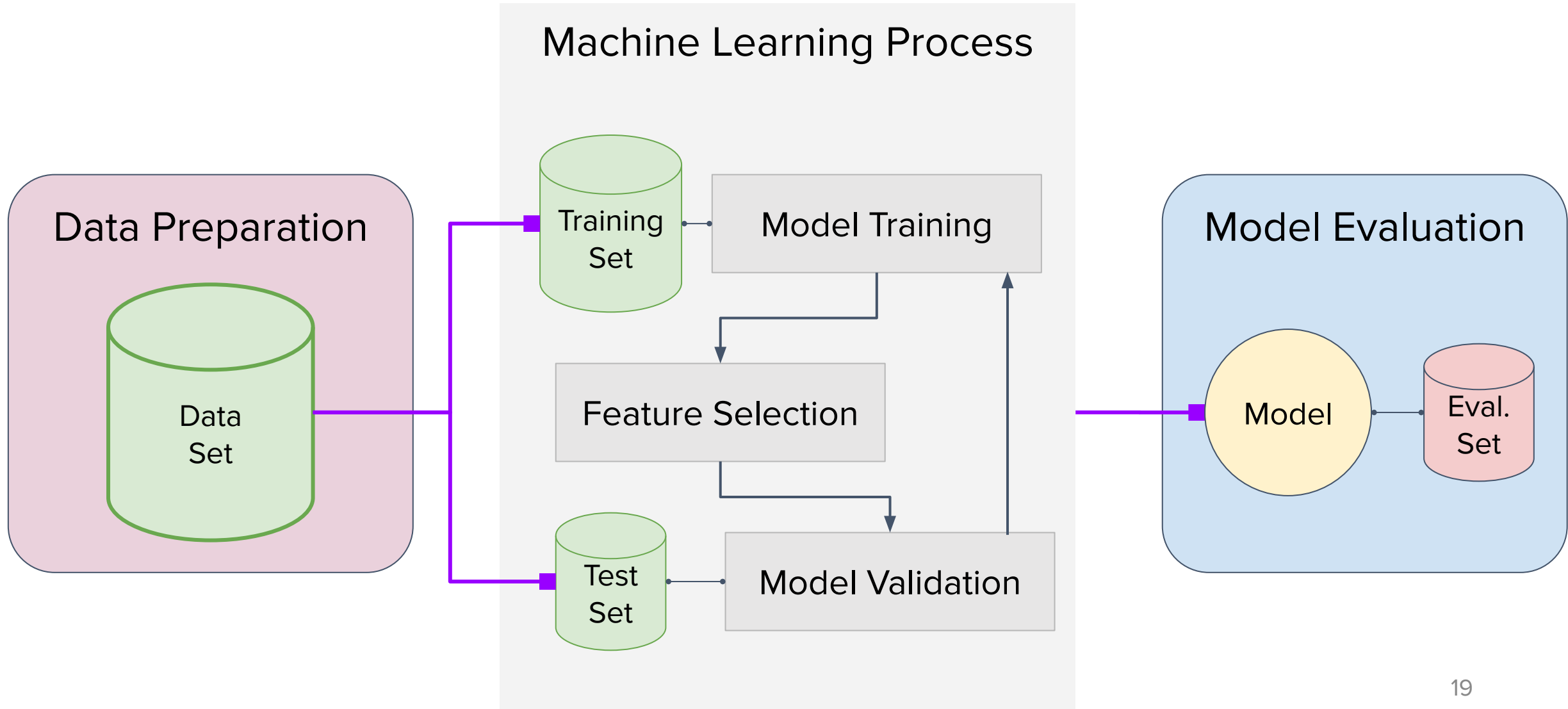




“Developing reservoir monthly inflow forecasts using artificial intelligence and climate phenomenon information”

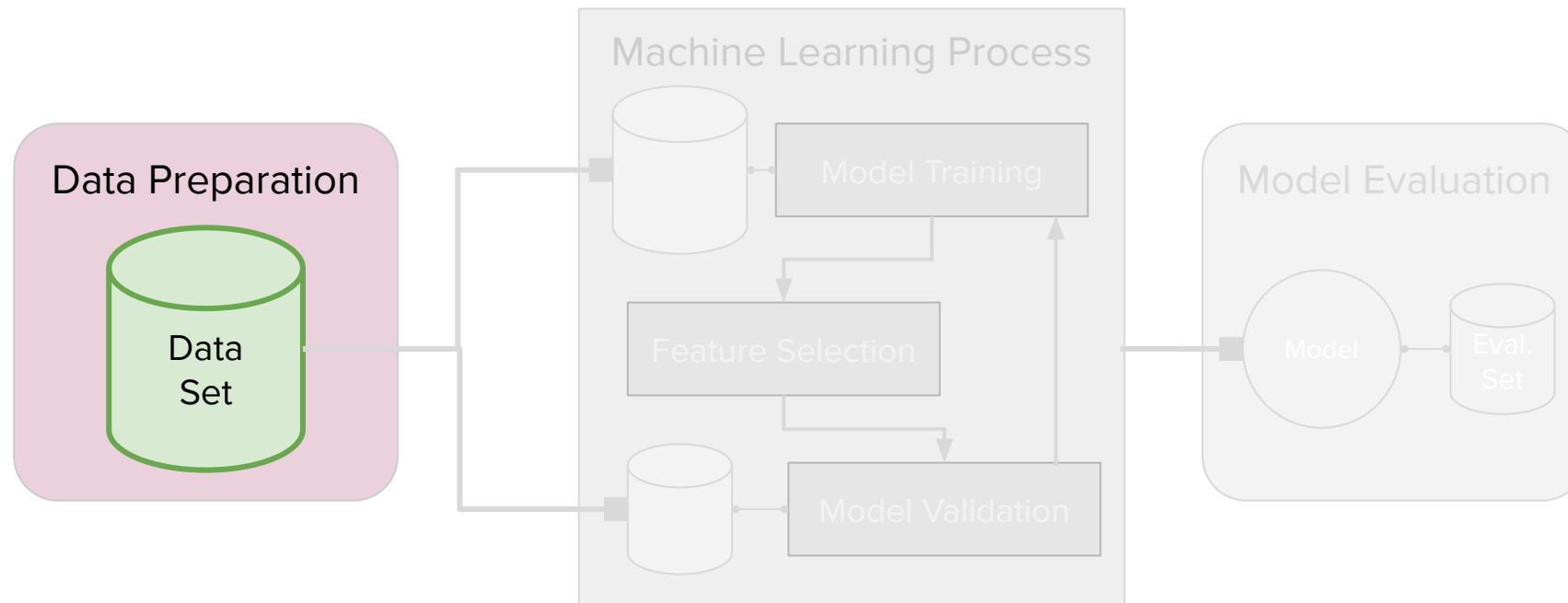
Ref: <https://doi.org/10.1002/2017WR020482>







- Data aggregation (if needed)
- Understanding your domain/data
- Data preprocessing





- What is your goal/objective? *Inflow at $t+3$ day*
(Response variables/ Outcome/ Dependent variable)
- What is your predictors? *Remaining variables*
(Independent variables)

Inflow_t3	Inflow_t	Highest Temperature	Year	...	Date
5.65	0.63	43	dry year	...	2018-12-13
3.94	3.67	40	dry year	...	2018-12-14
4.64	7.59	42	dry year	...	2018-12-15
8.63	5.65	42	dry year	...	2018-12-16
...



- Data is complete and clean --- We are happy :)
- Data is not always perfect i.e. incomplete, missing, error

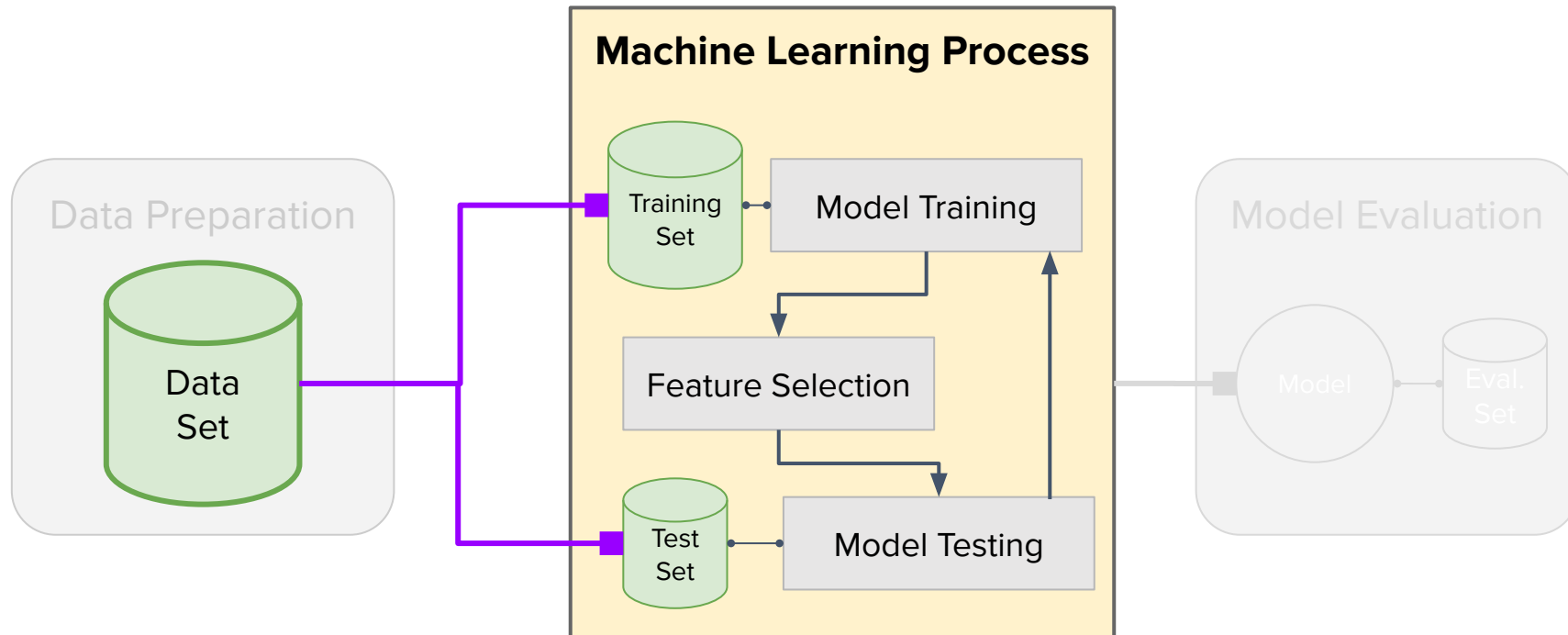
date	day	month	year	water.lvl	storage	inflow	extra.wate	adj.inflow	released	spilled	irrigation	evploss	pumped	end.storag	diff	adj.inflow2	avg.humidity	avg.pressu	avg.sealvlp	avg.temp
7/25/2008	25	7	2008	230.95	6334.41	13.13	0	13.13	12.68	0	0	0.45	0	6334.41	0	13.13	80	987.2	989.1	26.6
7/26/2008	26	7	2008	230.96	6335.87	11.9	0	11.9	9.99	0	0	0.45	0	6335.87	0	11.9				
7/27/2008	27	7	2008	230.98	6338.79	15.37	0	15.37	12	0	0	0.45	0	6338.79	0	15.37	62	988.2	990.1	29
7/28/2008	28	7	2008	230.98	6338.79	10.39	0	10.39	9.94	0	0	0.45	0	6338.79	0	10.39				
7/29/2008	29	7	2008	231.04	6347.57	19.25	0.01	19.24	10.01	0	0	0.45	0	6347.57	0	19.24				
7/30/2008	30	7	2008	231.06	6350.51	13.49	0.01	13.5	10.11	0	0	0.45	0	6350.51	0	13.5				
7/31/2008	31	7	2008	231.06	6350.51	11.07	0	11.07	10.62	0	0	0.45	0	6350.51	0	11.07	64	988.6	990.5	28.2
8/1/2008	1	8	2008	231.06	6350.51	14.34	0	14.34	13.91	0	0	0.43	0	6350.51	0	14.34	64	988.6	990.5	29.6
8/2/2008	2	8	2008	231.05	6349.04	9.29	0	9.29	10.33	0	0	0.43	0	6349.04	0	9.29	70	988.9	990.8	28
8/3/2008	3	8	2008	231.07	6351.98	14.23	0	14.23	10.86	0	0	0.43	0	6351.98	0	14.23	55	988.6	990.5	30.6
8/4/2008	4	8	2008	231.13	6360.82	19.32	0	19.32	10.05	0	0	0.43	0	6360.82	0	19.32				
8/5/2008	5	8	2008	231.17	6366.72	16.43	0	16.43	10.09	0	0	0.44	0	6366.72	0	16.43				
8/6/2008	6	8	2008	231.2	6371.16	15.92	0	15.92	11.04	0	0	0.44	0	6371.16	0	15.92				
8/7/2008	7	8	2008	231.25	6378.58	17.28	0	17.28	10.02	0	0	0.44	0.6	6378.58	0	17.28				
8/8/2008	8	8	2008	231.33	6380.5	22.28	0	22.28	9.93	0	0	0.44	0	6380.5	0	22.28				



- Data preprocessing usually consumes about 80% of the time spent on data analysis for the research project
 - Check out the missing values
 - Remove outliers if needed
 - Interpolate/**Impute data** if needed
 - Do we need to discretize variable?
 - Sometime, we need to generate more variables for predictions



- Part of data (70%-80%) are randomly chosen to be on a training set
- The remaining data are reserved for the test set





- Including all predictors are not guaranteed the best performance
- A large number of features make a model bulky, time-taking, and harder to implement in production.

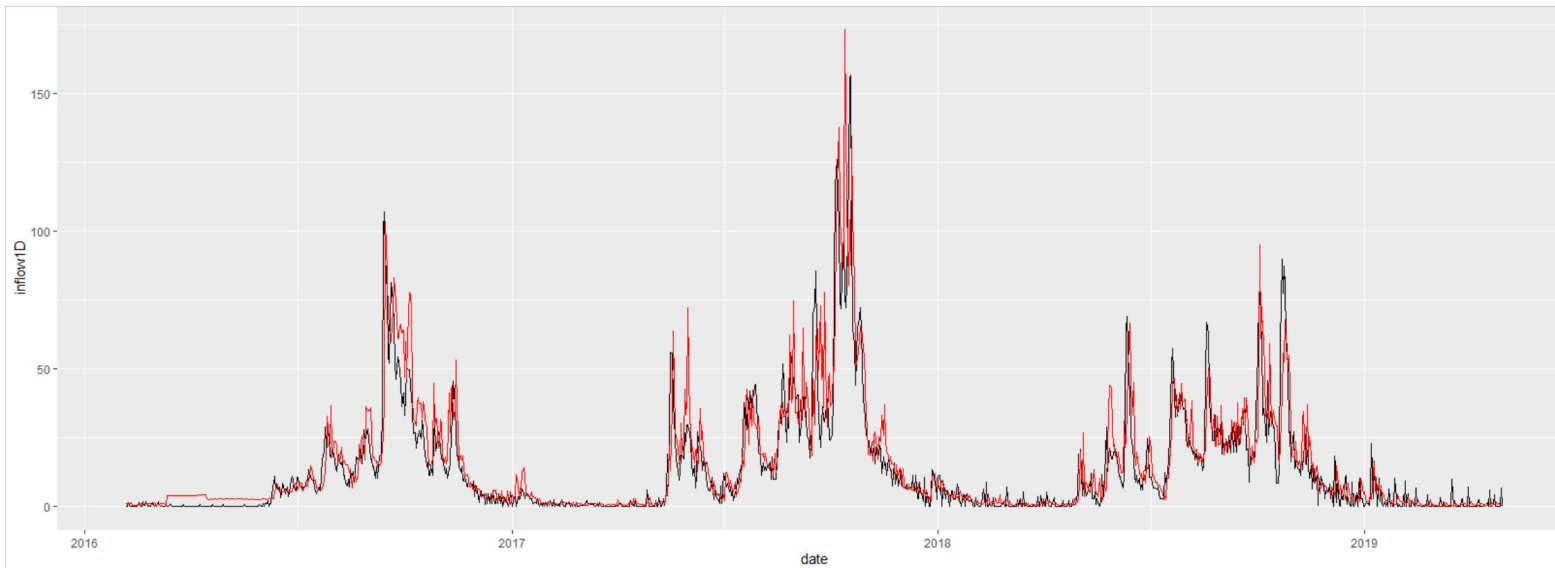
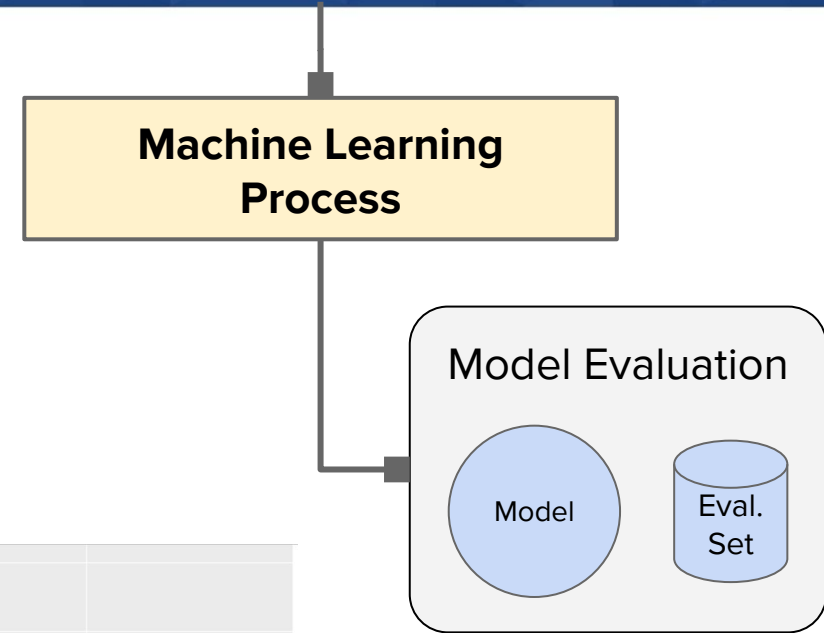
- Find the best set of predictors that generates the best model (*can be done during modeling*)
- Can be done **manually** or **statistically**



- Linear regression
- Logistic regression
- Naive Bayes Classifier
- Neural network/ Deep learning
- ARMA (Autoregressive Moving Average)
- ARIMA (Autoregressive Integrated Moving Average)
- Adaptive Neuro-Fuzzy Inference System (ANFIS)
- Support Vector Regression (SVR)
- Decision Tree
- Random Forest (RF)
- **Extreme Gradient Boosting (XGBoost)**
- ...



- Test with the untouched data set in the real situation
- May need model tuning if **overfitted (e.g. over-train the model)**

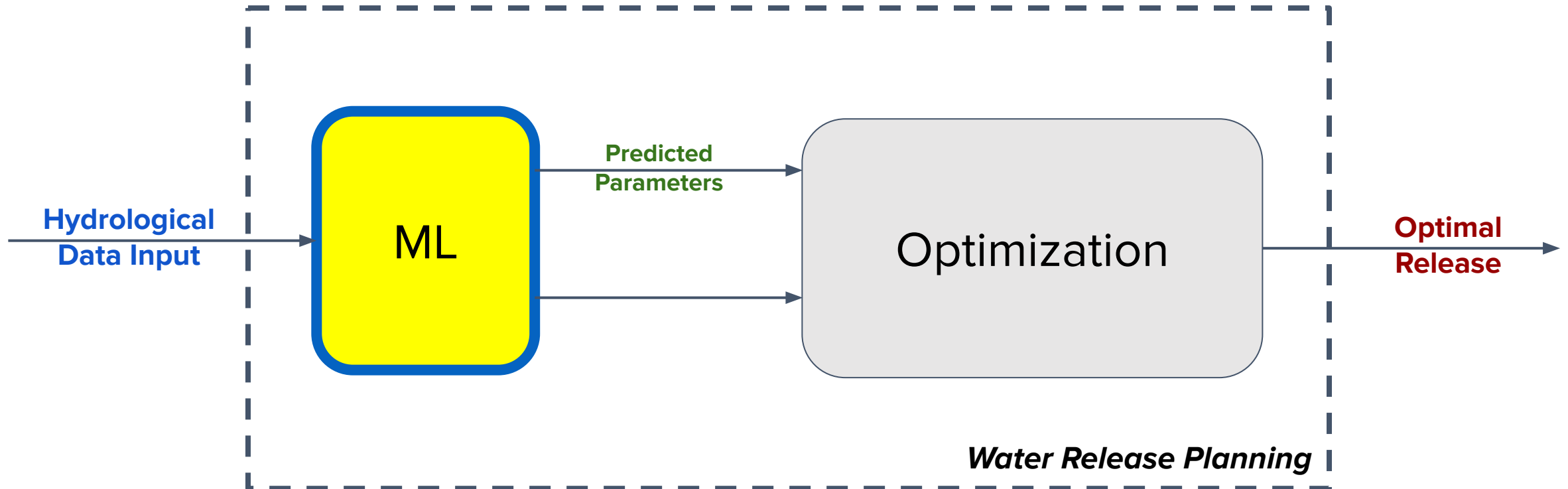




Mahidol University
Wisdom of the Land



Case Study: ML in Reservoir Management





How we predict/forecast inflow? or What affect inflow?

- Past inflow
- Precipitation
- Evaporation loss
- Humidity
- ...



< Sample_DataSet.csv (565.09 KB)

Detail Compact Column

About this file

This file does not have a description yet.

date	# day	# month	# year	# water.lvl	# storage	# inflow	# extra.water	# adj.inflow	# released
25Jul08 - 30Apr19	1 - 31	1 - 12	2008 - 2019	214 - 260	3.92k - 13.5k	0 - 311	0 - 0.01	0 - 311	0 - 64.5
7/25/2008	25	7	2008	230.95	6334.41	13.13	0	13.13	12.68
7/26/2008	26	7	2008	230.96	6335.87	11.9	0	11.9	9.99
7/27/2008	27	7	2008	230.98	6338.79	15.37	0	15.37	12
7/28/2008	28	7	2008	230.98	6338.79	10.39	0	10.39	9.94
7/29/2008	29	7	2008	231.04	6347.57	19.25	0.01	19.24	10.01
7/30/2008	30	7	2008	231.06	6350.51	13.49	0.01	13.5	10.11
7/31/2008	31	7	2008	231.06	6350.51	11.07	0	11.07	10.62
8/1/2008	1	8	2008	231.06	6350.51	14.34	0	14.34	13.91
8/2/2008	2	8	2008	231.05	6349.04	9.29	0	9.29	10.33
8/3/2008	3	8	2008	231.07	6351.98	14.23	0	14.23	10.86
8/4/2008	4	8	2008	231.13	6360.82	19.32	0	19.32	10.05

Daily Inflow



- R is a free software environment for
 - a wide variety of statistical libraries for data analysis (linear and nonlinear modelling, classical statistical tests, time-series analysis)
 - machine learning (classification, clustering, ...)
 - graphical techniques
- R is available as Free Software
- Download: <http://mirrors.psu.ac.th/pub/cran/>



RStudio interface showing R code and a scatter plot.

```
18 )
19
20 # Create a scatterplot between spi and avg_rainfall_jul with color g
21 ggplot(ds, aes(x = spi, y = avg_rainfall_jul, color=class)) +
22   geom_point() + # Show dots
23   geom_text(
24     label=ds$year, # Label on data
25     nudge_x = 0.25, nudge_y = 0.25, # shifts the text along x and y
26     check_overlap = T # Avoid overlapping of the text?
27   )
28
29 # Create a histogram
30 data = data.frame(value=rnorm(100))
31 ggplot(data, aes(x=value)) + geom_histogram()
32
33
34
```

Environment: Global Environment

Data: data (100 obs. of 1 variable)

Files: Plots Packages Help Viewer

class

- dry year
- wet year

Year	spi	avg_rainfall_jul	class
1998	1.0	600	wet year
1999	1.0	580	wet year
2001	-1.5	600	dry year
2010	1.8	580	wet year
1993	0.8	580	wet year
1989	1.2	580	wet year
2002	1.5	580	wet year
2004	1.5	540	wet year
2005	0.8	480	wet year
2009	1.2	480	wet year
1996	-0.5	380	dry year
1995	-0.2	330	dry year
1992	-1.0	300	dry year
1990	-1.2	260	dry year
2003	-0.8	260	dry year
1997	0.5	260	dry year
2000	2.2	260	wet year
2007	-1.0	220	dry year
1991	-0.8	220	dry year
1988	-1.2	200	dry year



“Explore and run machine learning code with Kaggle Notebooks, a cloud computational environment ...” -- Kaggle Notebook [[ref](#)]

← → ↻ kaggle.com/kjidapa/simple-inflow-prediction-of-a-dam

Search

Online platform for practicing R

Simple Inflow Prediction of a Dam
R notebook using data from [Reservoir and Weather data](#) · 18 views · 8d ago · Edit tags

Step 0: Prepare R packages

```
In [1]:  
# Step 0: REQUIRED: PACKAGES -- Run once at the beginning  
# -----  
library(imputeTS) # For na_kalman imputation  
library(RcppRoll) # For lag, lead, and average  
library(dplyr)    # For mutate (Data manipulation)  
library(xgboost)  
library(Matrix)  
library(ggplot2) # For ggplot visualization  
library(zoo)  
library(Metrics)
```

Version 2 of 2
Notebook
Step 0: Prepare R Packages
Step 1: Data Preparation, I.E.,...
Step 2: Training Models With XGBoost
Step 3: Visualization Of The Predicted Result
Input (1)



- **Part 0: Prepare your Kaggle Notebook**
 - **Set up Data Set**
 - **Set up R environment**
- Part I: Understanding your data Data Preparation
- Part II: Training the Model with XGBoost
- Part III: Visualization



1. Go to <https://www.kaggle.com/kjidapa/simple-inflow-prediction-of-a-dam>
2. Sign in with your email of preferences

The screenshot shows a web browser window displaying a Kaggle notebook titled "Simple Inflow Prediction of a Dam". The browser's address bar shows the URL <https://www.kaggle.com/kjidapa/simple-inflow-prediction-of-a-dam>. The Kaggle logo and navigation menu are visible on the left. The notebook content includes a search bar, a "Sign In" button (highlighted with a red box), and a "Register" button. Below the notebook title, there is a "Copy and Edit" button with a count of 2. The main content area shows "Step 0: Prepare R packages" with the following R code:

```
In [1]:  
# Step 0: REQUIRED: PACKAGES -- Run once at the beginning  
# -----  
library(imputeTS) # For na_kalman imputation  
library(RcppRoll) # For lag, lead, and average  
library(dplyr)    # For mutate (Data manipulation)  
library(xgboost)  
library(Matrix)  
library(ggplot2) # For ggplot visualization  
library(zoo)  
library(Metrics)
```

On the right side, there is a sidebar with "Version 2 of 2", "Notebook", and a list of steps: "Step 0: Prepare R Packages", "Step 1: Data Preparation, I.E.,...", "Step 2: Training Models With XGBoost", and "Step 3: Visualization Of The Predicted Result". A sign-in modal box is open on the right, highlighted with an orange border, showing options to "Sign in with Google", "Sign in with your email", "Sign in with Facebook", and "Sign in with Yahoo". A "No Account? Create one." link is also present. A red arrow points from the "Sign In" button on the page to the modal box.



3. Click “Copy and Edit”

You can view your own code and modify the code if you needed

The screenshot shows a web browser window displaying a Kaggle notebook titled "Simple Inflow Prediction of a Dam". The browser address bar shows the URL "kaggle.com/kjidapa/simple-inflow-prediction-of-a-dam". The notebook interface includes a search bar, a navigation menu on the left, and a main content area. The notebook title is "Simple Inflow Prediction of a Dam" with a subtitle "R notebook using data from [Private Datasource] - 23 views - 8d ago". A red box highlights the "Copy and Edit" button, which has a "2" next to it. The notebook content shows "Step 0: Prepare R packages" with the following code:

```
In [1]:  
# Step 0: REQUIRED: PACKAGES -- Run once at the beginning  
# -----  
library(imputeTS) # For na_kalman imputation  
library(RcppRoll) # For lag, lead, and average  
library(dplyr)    # For mutate (Data manipulation)  
library(xgboost)  
library(Matrix)  
library(ggplot2) # For ggplot visualization  
library(zoo)  
library(Metrics)
```

On the right side of the notebook, there is a sidebar with the following items:

- Version 2 of 2
- Notebook
- Step 0: Prepare R Packages
- Step 1: Data Preparation, I.E.,...
- Step 2: Training Models With XGBoost
- Step 3: Visualization Of The Predicted Result



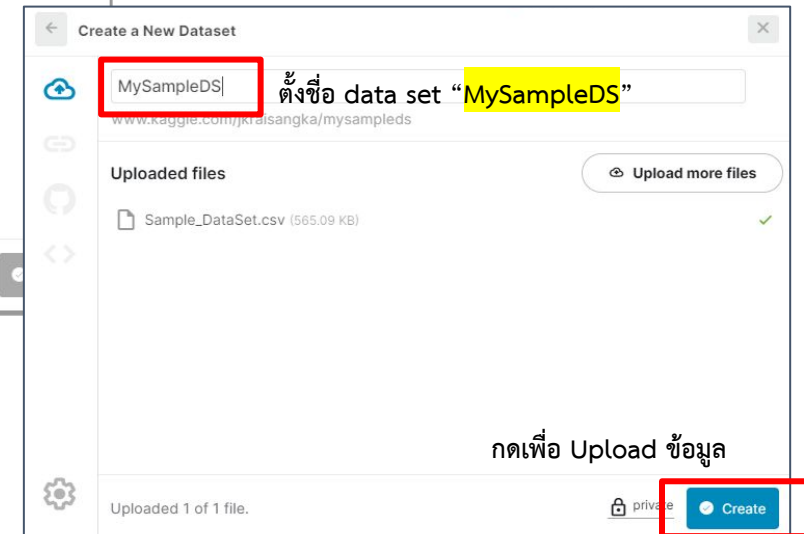
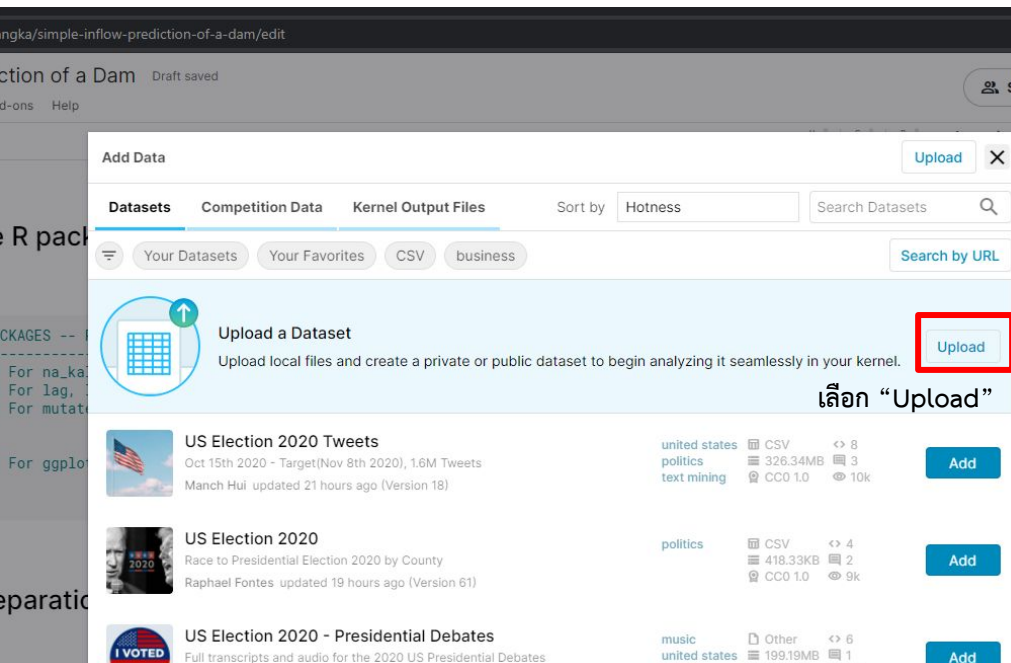
4. Go to: <http://bit.ly/simpleDamDS> and download the Data set

The screenshot shows a dataset page for "Reservoir and Weather data" by KJidapa. The page includes a "Data Explorer" section for "Sample_DataSet.csv (565.09 KB)". In the top right corner of the data explorer, there is a download icon (a downward arrow) which is highlighted with a red square. Below the data explorer, there are several bar charts representing different data points: date (25Jul08 to 30Apr19), # day (1 to 31), # month (1 to 12), # year (2008 to 2019), and # water.lvl (214).

คลิกปุ่มเพื่อดาวน์โหลด "MySampleDS.csv"



5. Go back to your Kaggle notebook and upload the data set by “Add data”





Simple Inflow Prediction of a Dam Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All

Draft Session (41m) H D D C P U R A M

“MySampleDS” จะอยู่ใน Input

Step 0: Prepare R packages

```
[1]: # Step 0: REQUIRED: PACKAGES -- Run once at the beginning
# -----
library(imputeTS) # For na_kalman imputation
library(RcppRoll) # For lag, lead, and average
library(dplyr) # For mutate (Data manipulation)
library(xgboost)
library(Matrix)
library(ggplot2) # For ggplot visualization
library(zoo)
library(Metrics)
```

Language ต้องเปลี่ยนเป็น R

input (565.09 KB)
mysampled

output
/kaggle/working

Settings

Language R

Environment Preferences

Accelerator [Requires phone verification](#)

Internet [Requires phone verification](#)

Code Help

Find Code Help

Search for examples of how to do things

Step 1: Data preparation, i.e., cleaning the data, imputing the missing values



Demo: Kaggle Notebook





กดเพื่อรันโค้ดในกล่องนี้



```
# 2. Select some variables
selectedCol <- c( # Date and time
  "date", "year", "month", "day",
  # Dam data
  "water.lvl", "adj.inflow", "released",
  "spilled", "irrigation", "evploss",
  # Weather data
  "avg.humidity", "avg.pressure",
  "avg.temp", "avg.pcp")
myDS <- myDS[,selectedCol]
print("Selected some variables to be analyzed")
head(myDS) # Show the top records of our data set
```

Output ของโค้ด

[1] "Selected some variables to be analyzed"

A data.frame: 6 × 14

	date	year	month	day	water.lvl	adj.inflow	released	spilled	irrigation	evploss	avg.humidity	avg.pressure	avg.temp	avg.pcp
	<fct>	<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	7/25/2008	2008	7	25	230.95	13.13	12.68	0	0	0.45	80	987.2	26.6	15.8
2	7/26/2008	2008	7	26	230.96	11.90	9.99	0	0	0.45	NA	NA	NA	NA
3	7/27/2008	2008	7	27	230.98	15.37	12.00	0	0	0.45	62	988.2	29.0	0.2
4	7/28/2008	2008	7	28	230.98	10.39	9.94	0	0	0.45	NA	NA	NA	NA
5	7/29/2008	2008	7	29	231.04	19.24	10.01	0	0	0.45	NA	NA	NA	NA
6	7/30/2008	2008	7	30	231.06	13.50	10.11	0	0	0.45	NA	NA	NA	NA



- Part 0: Prepare your Kaggle Notebook
 - Set up Data Set
 - Set up R environment
- **Part I: Understanding your data Data Preparation**
- Part II: Training the Model with XGBoost
- Part III: Visualization



List, Look, and Clean

```
In [ ]:  
  
# 2. Select some variables  
selectedCol <- c( # Date and time  
  "date", "year", "month", "day",  
  # Dam data  
  "water.lvl", "adj.inflow", "released",  
  "spilled", "irrigation", "evploss",  
  # Weather data  
  "avg.humidity", "avg.pressure",  
  "avg.temp", "avg.pcp")  
myDS <- myDS[,selectedCol]  
print("Selected some variables to be analyzed")  
head(myDS) # Show the top records of our data set
```

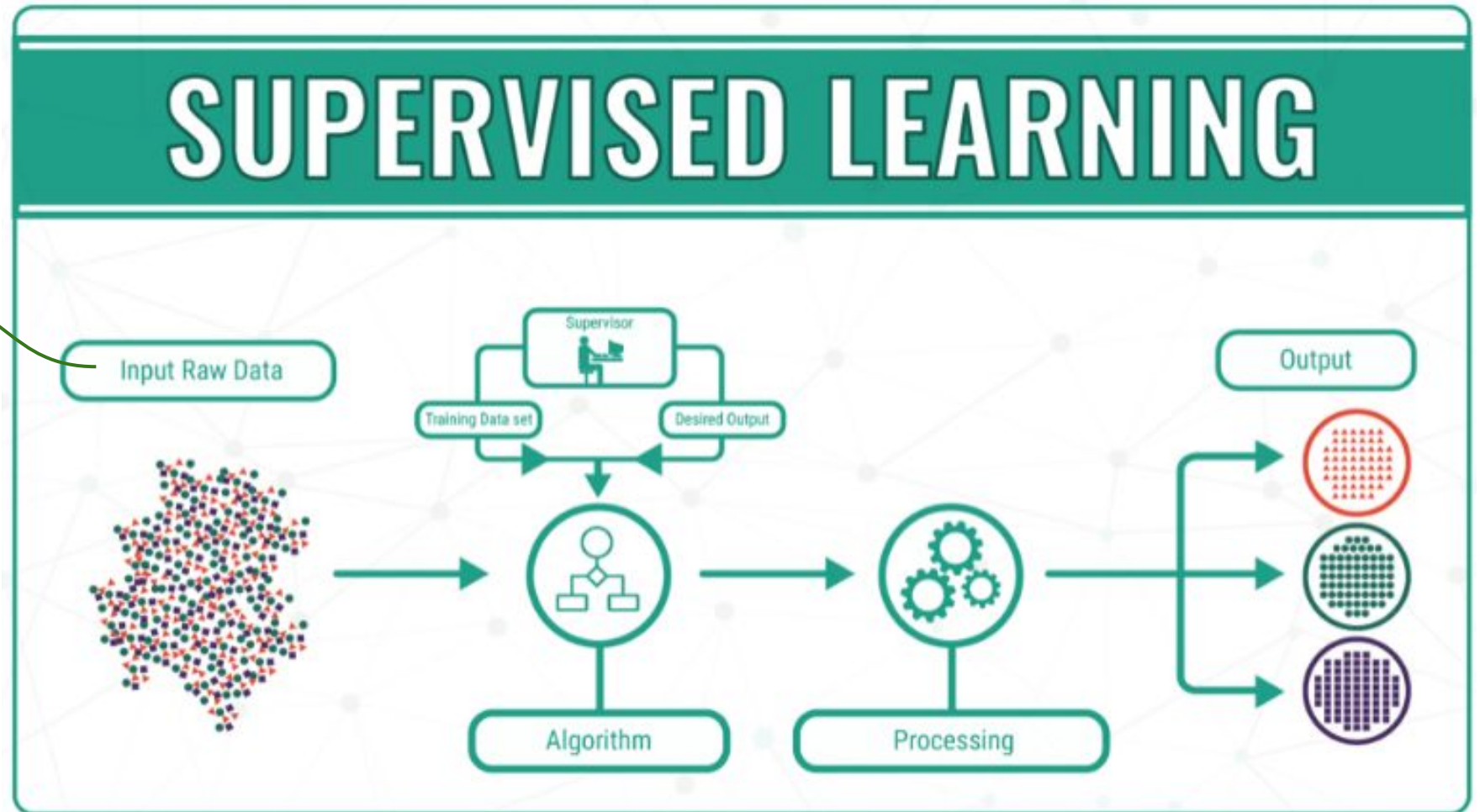


- Part 0: Prepare your Kaggle Notebook
 - Set up Data Set
 - Set up R environment
- Part I: Understanding your data Data Preparation
- **Part II: Training the Model with XGBoost**
- Part III: Visualization



SUPERVISED LEARNING

Data has its desired output, i.e., class or target

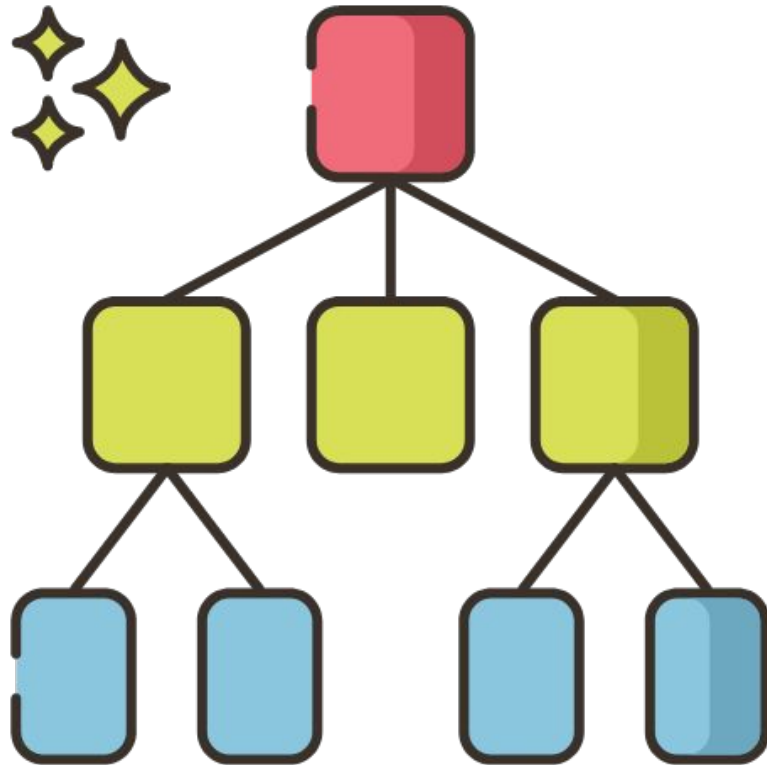




- **Extreme Gradient Boosting (XGBoost)**
- an approach where new models are **created that predict the residuals or errors of prior models** and then added together to make the final prediction.
- Suitable for small-to-medium structured/tabular data
-
- This approach supports both **regression** and classification predictive modeling problems.



Target inflow($t+3$) = 4.00



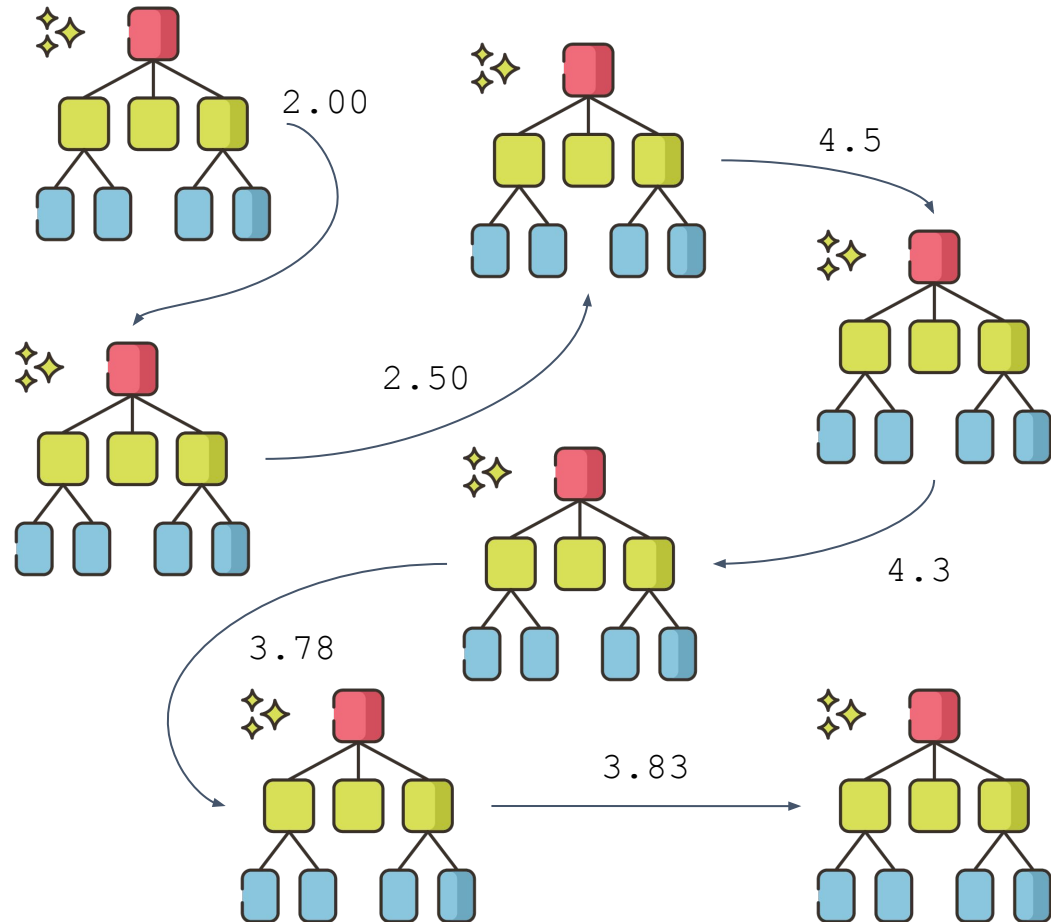
(Regression) Decision Tree

- The predicted outcome can be considered a real number (e.g. inflow at $t+3$).
- Each node is a predictor
- One tree gave one predicted output

→ Predicted inflow($t+3$) = 2



Target inflow($t+3$) = 4.00



Ensembled Decision Tree - Gradient Boosting

It uses gradient descent algorithm which can optimize **any differentiable loss function**. An ensemble of trees are **built one by one** and individual trees are summed sequentially. Next tree tries to **recover the loss** (difference between actual and predicted values), e.g. **MSE: mean square error**

Ref: <https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9>

Predicted inflow($t+3$) = 3.9



- Derived from Decision Tree Ensemble plus **training loss** and **regularization**

Training loss ensures the model fits the training data well

+

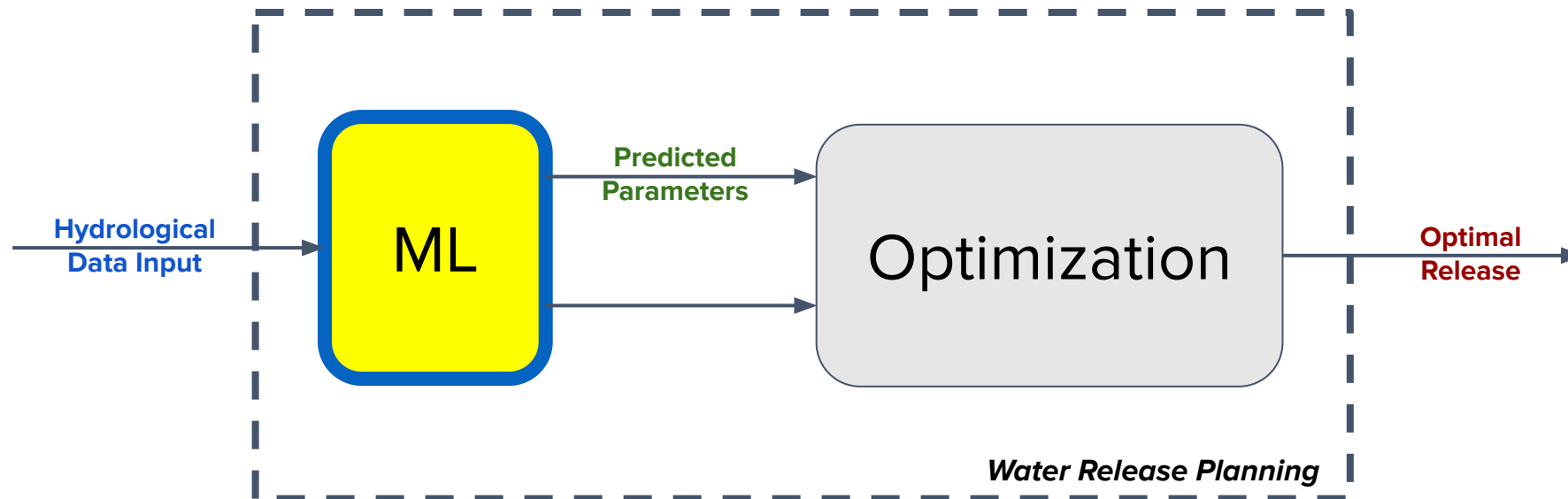
Regularization controls the complexity of the model, which helps us to avoid overfitting.

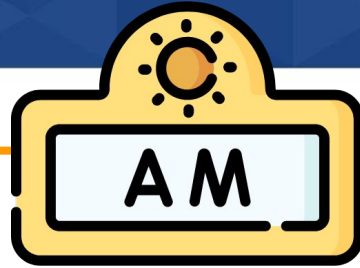


- Part 0: Prepare your Kaggle Notebook
 - Set up Data Set
 - Set up R environment
- Part I: Understanding your data Data Preparation
- Part II: Training the Model with XGBoost
- **Part III: Visualization**



- ML can be used as alternative tools for reservoir operation
- Building a good ML model requires
 - Understanding your data
 - Preprocessing
 - Apply a good techniques
 - Validate your model





Basic Machine Learning Concept

+

Building a model with R



*Dr. Wudhichart
(Wud)
13:00 - 16:00*

Optimization

+

Building a model with MiniZinc



Constraint Programming with Water Resources Engineering: Modelling and Tool

SIP1 Workshop

Wudhichart Sawangphol

Faculty of ICT, Mahidol University

Nov 13, 2020

- 1 General Introduction
 - Applications
 - Sport Scheduling
 - Hong Kong International Airport (HKIA)
 - Port of Singapore
 - More Example Applications
 - CP Solving: Summary
 - CP software
- 2 MiniZinc
 - Basic Modelling in MiniZinc
 - Complex Models
 - Predicates and Functions
- 3 Other Constraint Modelling Languages
- 4 MiniZinc Software
- 5 Workshop
- 6 References

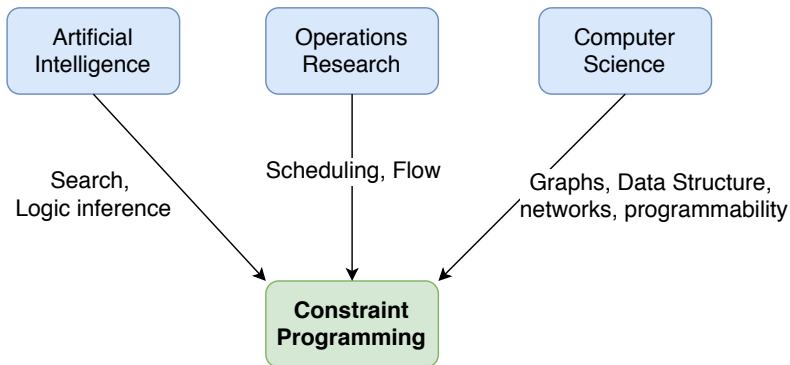
- 1 General Introduction
 - Applications
 - Sport Scheduling
 - Hong Kong International Airport (HKIA)
 - Port of Singapore
 - More Example Applications
 - CP Solving: Summary
 - CP software
- 2 MiniZinc
 - Basic Modelling in MiniZinc
 - Complex Models
 - Predicates and Functions
- 3 Other Constraint Modelling Languages
- 4 MiniZinc Software
- 5 Workshop
- 6 References

What is Constraint Programming?



Sudoku is Constraint Programming

Constraint Programming (CP) Overview



- 1970s: Image processing applications in AI; Search+qualitative inference
- 1980s: Logic Programming (Prolog); Search + logical inference
- 1989: CHIP System; Constraint Logic Programming
- 1990s: Constraint Programming; Industrial Solvers (ILOG, Eclipse,...)
1994: Advanced inference for alldifferent and resource scheduling
- 2000s: Global constraints; integrated methods; modeling languages
- 2005: Gecode [1] CP solver is established
- 2006: CISCO Systems acquires Eclipse CLP solver
- 2007: Minizinc: a standard CP modelling language [2] is established
- 2009: IBM acquires ILOG CP Solver & Cplex

- Some sports are *Round-robin scheduling problems*.
- Examples: USA basketball leagues and European football leagues
- Schedule of 1997/1998 Atlantic Coast Conference (ACC) basketball league (9 universities)
 - ▶ all 179 solutions were found in 24h using enumeration and integer linear programming [3]
 - ▶ all 179 solutions were found in less than a minute using constraint programming [4]



Hong Kong International Airport (HKIA) [5]

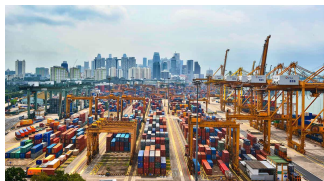


- HKIA serviced around 100,000 passengers daily.
- Main responsibility is on the Hong Kong Airport Authority (AA)
- The AI system called *Stand Allocation System (SAS)* was developed using CP.
- The system supports concurrent use by multiple operators in non-stop 24 hours-a-day operations, since HKIA is a 24-hour airport.

- Objectives:
 - ▶ To be as fair as possible to all airlines and handling agents, while making efficient and safe use of airport resources
 - ▶ To provide better service and comfort to passengers. For example, the system need to assign an aircraft to the gate that is close to the immigration counters of passengers.
- SAS can produce allocation plan in **3 mins** and react problem solving in **5 seconds**



- One of the world's largest container transshipment hubs
- A network of 200 shipping lines, which is connected to 600 ports in 123 countries around the world
- Challenges: Yard allocation and loading plans under various operational and safety requirements
- Solution: Yard planning system, based on constraint programming (ILOG)



More Example Applications

- Railroad optimization
- Rail driver allocation
- Supply chain

The solution process of CP interleaves

- **Search**: enumerating all possible assignments. The size of search tree may be exponential. Therefore, *domain filtering* and *constraint propagation* are needed to reduce the size of search tree.
- **Domain filtering**: Removing inconsistent values from the domains of the variables regarding individual constraints
- **Constraint propagation**: propagating the filtered domains through the constraints by re-evaluating them until there are no more changes in the domains

- As can be seen, the CP software can be divided into 2 groups:
 - ▶ Constraint Solver
 - ▶ Constraint Modelling system

- There are many Constraint Solvers available.
- They are commercial, non-commercial, and open source.
- A constraint solver has to
 - ▶ [find a solution](#) for CSP
 - ▶ or [prove that no solution](#) exists

- Commercial Constraint Solvers
 - ▶ ILOG Solver, C++
 - ▶ CPLEX, C++
 - ▶ Kalis (Artelys), C++
- Non-commercial Constraint Solvers
 - ▶ GeCode: Generic Constraint development Environment, C++
 - ▶ Chuffed, C++
 - ▶ Minion, C++
 - ▶ Ipopt, C++
 - ▶ Google OR-tools, C++
 - ▶ Choco, Java
 - ▶ Jacop, Java
 - ▶ FiPy, Python

- Commercial
 - ▶ ILOG CPLEX Optimization Studio
 - ▶ Xpress
 - ▶ AIMMS
- Non-commercial
 - ▶ MiniZinc
 - ▶ Numberjack
 - ▶ Minion

- 1 General Introduction
 - Applications
 - Sport Scheduling
 - Hong Kong International Airport (HKIA)
 - Port of Singapore
 - More Example Applications
 - CP Solving: Summary
 - CP software
- 2 MiniZinc
 - Basic Modelling in MiniZinc
 - Complex Models
 - Predicates and Functions
- 3 Other Constraint Modelling Languages
- 4 MiniZinc Software
- 5 Workshop
- 6 References

- MiniZinc is a medium-level declarative modelling language [2].
- MiniZinc has been developed as a standard modelling language for modelling Constraint Programming problems.
- MiniZinc is solver-independent.
- MiniZinc supports the separation of the model and the data. A couple (model, data) is called an **instance**.

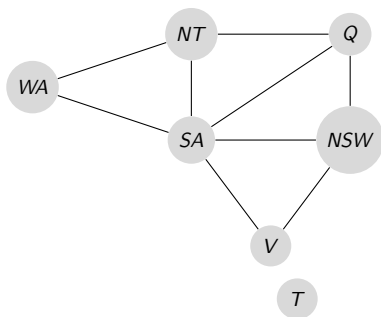
- A MiniZinc instance is solved by:
 - ① Compiling the instance into a FlatZinc model.
 - ② Solving the FlatZinc model using one of many existing back-end CP solvers.

Recall: Australia Map Coloring



- Problems: Coloring each region with one of 3 colors (red, green, blue) such that adjacent regions must have different colors
- From Australia map, there are 7 regions: Western Australia (WA), Northern Territory (NT), South Australia (SA), Queensland (Q), New South Wales (NSW), Victoria (V), and Tasmania (T)

Recall: CP Model



- **Variables:** WA, NT, SA, Q, NSW, V, T
- **Domains:** D_i where $i \in \{WA, NT, SA, Q, NSW, V, T\} = \{\text{red, green, blue}\}$
- **Constraints:** adjacent regions must have different colors
 $WA \neq NT, WA \neq SA, NT \neq SA, NT \neq Q, SA \neq Q, SA \neq NSW, SA \neq V, Q \neq NSW, NSW \neq V$

- The below model is an example MiniZinc model of Map colouring problems.

```
1 % Colouring Australia using nc colours
2 int: nc = 3;
3
4 var 1..nc: wa;   var 1..nc: nt;   var 1..nc: sa;   var 1..nc: q;
5 var 1..nc: nsw; var 1..nc: v;   var 1..nc: t;
6
7 constraint wa != nt;
8 constraint wa != sa;
9 constraint nt != sa;
10 constraint nt != q;
11 constraint sa != q;
12 constraint sa != nsw;
13 constraint sa != v;
14 constraint q != nsw;
15 constraint nsw != v;
16 solve satisfy;
17
18 output ["wa=", show(wa), "\t nt=", show(nt),
19         "\t sa=", show(sa), "\n", "q=", show(q),
20         "\t nsw=", show(nsw), "\t v=", show(v), "\n",
21         "t=", show(t), "\n"];
```

- A MiniZinc model is comprised of the following components:
 - ▶ Variable declarations
 - ★ Parameter declarations
 - ★ Decision Variable declarations
 - ▶ Constraints
 - ▶ Objective
 - ▶ Output command
 - ▶ Predicate declarations
 - ▶ Function declarations

```
1 int: nc = 3;
```

- This example is **Parameter Declaration**.
- In this example, **nc** is the number of colours to be used.
- Parameters are similar to (constant) variables in most programming languages.
- They must be declared and given type. In this case, the type is **int**.
- The assignment (**nc = 3**) can be included in the declaration or be a separate assignment statement.

```
1 int: nc;  
2 nc = 3;
```

Note that a parameter can only be given a *single value*.

- MiniZinc is strongly typed. The following types offered in MiniZinc:
 - ▶ `int`: integer
 - ▶ `bool`: Boolean
 - ▶ `float`: floating-point number
 - ▶ `string`: string of characters
 - ▶ `set of ty`: set of elements of type `ty`, which can be `int`, `bool`, `float`, or `string`
 - ▶ `array[r] of ty`: possibly multidimensional array of elements of type `ty` (anything except array). The range `r` is an integer interval `l..u` in each dimension.

```
1 var 1..nc: wa;   var 1..nc: nt;   var 1..nc: sa;   var 1..nc: q;  
2 var 1..nc: nsw; var 1..nc: v;   var 1..nc: t;
```

- This example is [Decision Variable Declaration](#).
- Unlike parameters and variables in a standard programming language, *the modeller does not need to give them a value*.
 - ▶ The values of decision variable will be assigned by solving system if the assignments satisfy the constraints in a constraint model.
- In this example, a decision variable is associated with each region, [wa](#), [nt](#), [sa](#), [q](#), [nsw](#), [v](#), and [t](#)
- These decision variable stand for the (unknown) colour to be used to fill the region.
- Each decision variable need to be given the set of possible values, which is called [variable's domain](#).
 - ▶ This can be given as part of the variable declaration and the type of the decision variable is inferred from the type of the values in the domain.

Decision Variable Declarations (cont.)



```
1 var 1..nc: wa;   var 1..nc: nt;   var 1..nc: sa;   var 1..nc: q;  
2 var 1..nc: nsw; var 1..nc: v;   var 1..nc: t;
```

- In this example, each decision variable is given the domain `1..nc`, which is an integer range expression indicating the set $\{1, 2, \dots, nc\}$ using the `var` declaration.

- Decision Variables are declared by using prefix `var` to the type or domain.
- The decision variable types are:
 - ▶ `int`: integer
 - ▶ `bool`: Boolean
 - ▶ `float`: floating-point number
 - ▶ `sets`
 - ▶ `arrays` whose elements are decision variables.

```
1 int: n = 5;  
2 int M;  
3 M= 20;  
4 set of int: prime = {2, 3, 5, 7}  
5 var int: X;  
6 var 0..59: mins = X + n;  
7 set of int: N = 1..n;  
8 array[N] of var float: St1;  
9 array[N] of float: St = [10.50,20.00,3.56,4.36,7.25];
```

- All parameters must be given fixed values.
- Variables can be constrained at declaration.

Identifiers

Identifiers are used to name parameters and variables. They are sequences of lower and uppercase alphabetic characters, digits and the underscore "_" character. They must start with a alphabetic character. Thus, myvarname_1 is a valid identifier. MiniZinc (and Zinc) keywords are not allowed to be used as identifier names.

Relational Operators

MiniZinc provides the relational operators: equal ($=$ or $==$), not equal (\neq), strictly less than ($<$), strictly greater than ($>$), less than or equal to (\leq), and greater than or equal to (\geq).

```
1 constraint wa != nt;  
2 constraint wa != sa;  
3 constraint nt != sa;  
4 constraint nt != q;  
5 constraint sa != q;  
6 constraint sa != nsw;  
7 constraint sa != v;  
8 constraint q != nsw;  
9 constraint nsw != v;
```

- This example is **Constraints**.
- These specify the Boolean expressions that the decision variables must satisfy to be a valid solution to the model.
- In this example, it contains *not equal constraints* between the decision variables.

- A constraint is the **constraint** keyword followed by a Boolean expression that must be true in any solution.

```
1 constraint x < y;  
2 constraint sum(x) < 10;  
3 constraint forall(i in N)(  
4     St1[i] >= 0  
5 );
```

```
1 solve satisfy;
```

- This example is **Objective**.
- In this example, it is a *satisfaction* problem.
 - ▶ This objective is to find a values for the decision variables that satisfies the constraints,

- The objective is defined using `solve` keyword:
 - ▶ `solve satisfy;`
The problem is a *satisfaction* problem.
 - ▶ `solve minimize x;`
The objective is to *minimize* the value of variable x . x can be arithmetic expression.
 - ▶ `solve maximize x;`
The objective is to *maximize* the value of variable x . x can be arithmetic expression.
- Note that MiniZinc does not support multi-objective optimisation directly. Multiple objectives must be aggregated as a weighted sum.


```
1 output ["wa=", show(wa), "\t nt=", show(nt),  
2         "\t sa=", show(sa), "\n", "q=", show(q),  
3         "\t nsw=", show(nsw), "\t v=", show(v), "\n",  
4         "t=", show(t), "\n"];
```

- This example is **Output**.
- This tells MiniZinc what to print when the model has been executed and a solution is found.

- The `output` keyword describes what must be printed upon finding a solution.
- The `output` keyword is followed by a list of strings.

```
1 output [show(x)];  
2 output ["solution:"] ++ [if x[i] > 0 then show(x[i]) ++ ", " else "_", " endif | i in  
  1..5];
```

- The function `show` returns a *string* representing the value of the given expression.
- The operator `++` concatenates two strings or two lists.

Default Output

A MiniZinc model with no output will output a line for each decision variable with its value, unless it is assigned an expression on its declaration.

Problem Description:

- We are going to bake some cakes for sales in a festival. There are 2 types of cakes that we plan; 1. banana cake and 2. chocolate cake.
 - ▶ Banana cake recipes: 250g of self-raising flour, 2 mashed bananas, 75g sugar and 100g of butter.
 - ▶ Chocolate cake recipes: 200g of self-raising flour, 75g of cocoa, 150g sugar and 150g of butter.
- We sell a chocolate cake for \$4.50 and a banana cake for \$4.00.
- we have 4kg self-raising flour, 6 bananas, 2kg of sugar, 500g of butter and 500g of cocoa.
- **The objective is to find how many of each sort of cake should we bake for the festival to maximise the profit.**

Example Model:

```
1 var 0..100: b; % no. of banana cakes
2 var 0..100: c; % no. of chocolate cakes
3
4 % flour: a banana cake uses 250g, a chocolate cake uses 200g, we have 4000g
5 constraint 250*b + 200*c <= 4000;
6 % bananas: a banana cake uses 2 bananas, we have 6 bananas
7 constraint 2*b <= 6;
8 % sugar: a banana cake uses 75g, a chocolate cake uses 150g, we have 2000g
9 constraint 75*b + 150*c <= 2000;
10 % butter: a banana cake uses 100g, a chocolate cake uses 150g, we have 2000g
11 constraint 100*b + 150*c <= 500;
12 % cocoa: a chocolate cake uses 75g, we have 500g
13 constraint 75*c <= 500;
14
15 % maximize our profit
16 solve maximize 400*b + 450*c;
17
18 output ["no. of banana cakes = ", show(b), "\n",
19         "no. of chocolate cakes = ", show(c), "\n"];
```

The statement `solve maximize 400*b + 450*c;` is used to find a maximum profit. The output:

```
1 no. of banana cakes = 2
2 no. of chocolate cakes = 2
3 -----
4 =====
```


Integer Arithmetic Operators

MiniZinc provides the standard integer arithmetic operators. Addition (+), subtraction (-), multiplication (*), integer division (`div`) and integer modulus (`mod`). It also provides + and - as unary operators.

MiniZinc provides standard integer functions for absolute value (`abs`) and power function (`pow`).

Float Arithmetic Operators

MiniZinc provides the standard floating point arithmetic operators: addition (+), subtraction (-), multiplication (*) and floating point division (/). It also provides + and - as unary operators.

MiniZinc does not automatically coerce integers to floating point numbers. The built-in function `int2float` can be used for this purpose.

MiniZinc provides in addition the following floating point functions: absolute value (`abs`), square root (`sqrt`), natural logarithm (`ln`), logarithm base 2 (`log2`), logarithm base 10 (`log10`), exponentiation of e (`exp`), sine (`sin`), cosine (`cos`), tangent (`tan`), arcsine (`asin`), arccosine (`acos`), arctangent (`atan`), hyperbolic sine (`sinh`), hyperbolic cosine (`cosh`), hyperbolic tangent (`tanh`), hyperbolic arcsine (`asinh`), hyperbolic arccosine (`acosh`), hyperbolic arctangent (`atanh`), and power (`pow`) which is the only binary function, the rest are unary.

Data files:

- MiniZinc also support [Data files](#).
- MiniZinc allows the use of data files to set the value of parameters declared in the original model.
- Data files must have the file extension [.dzn](#) to indicate a MiniZinc data file and a model can be run with any number of data files (though a variable/parameter can only be assigned a value in one file).

Example Model with Data file:

```
1 int: flour; %no. grams of flour available
2 int: banana; %no. of bananas available
3 int: sugar; %no. grams of sugar available
4 int: butter; %no. grams of butter available
5 int: cocoa; %no. grams of cocoa available
6
7 var 0..100: b; % no. of banana cakes
8 var 0..100: c; % no. of chocolate cakes
9
10 % flour: a banana cake uses 250g, a chocolate cake uses 200g, we have 4000g
11 constraint 250*b + 200*c <= flour;
12 % bananas: a banana cake uses 2 bananas, we have 6 bananas
13 constraint 2*b <= banana;
14 % sugar: a banana cake uses 75g, a chocolate cake uses 150g, we have 2000g
15 constraint 75*b + 150*c <= sugar;
16 % butter: a banana cake uses 100g, a chocolate cake uses 150g, we have 2000g
17 constraint 100*b + 150*c <= butter;
18 % cocoa: a chocolate cake uses 75g, we have 500g
19 constraint 75*c <= cocoa;
20
21 % maximize our profit
22 solve maximize 400*b + 450*c;
23
24 output ["no. of banana cakes = ", show(b), "\n",
25         "no. of chocolate cakes = ", show(c), "\n"];
```

```
1 int: flour; %no. grams of flour available
2 int: banana; %no. of bananas available
3 int: sugar; %no. grams of sugar available
4 int: butter; %no. grams of butter available
5 int: cocoa; %no. grams of cocoa available
```

This part of model is parameters, which can be assigned values through data files.

The examples of data files:

```
1 flour = 4000;
2 banana = 6;
3 sugar = 2000;
4 butter = 500;
5 cocoa = 500;
```

```
1 flour = 8000;
2 banana = 11;
3 sugar = 3000;
4 butter = 1500;
5 cocoa = 800;
```


In this section, the array and set data structures, enumerated types and more complex constraints will be introduced.

Problem Description:

- We are going to organize the garden banquet.
- We have 2 type of tables: 1. VIP and 2. Normal.
 - ▶ VIP table:
 - ★ It has a capacity of 18.
 - ★ This table will be served with Prawn Tom Yum, Fried Sea Bass, and Roasted Duck
 - ★ Each dish has different satisfaction and capacity
 - ▶ Normal table:
 - ★ It has a capacity of 70.
 - ★ This table will be served with Hainanese Chicken, Fried Rice with egg, Isan Sausage, BBQ Pork and Soup
 - ★ Each dish has different satisfaction and capacity
- Every dish needs to be served.
- The objective is to **maximise** the satisfaction regarding capacity constraint.

Complex Models: Banquet problem (cont.)



VIP Table:

			
Satisfaction	29	19	8
Capacity	8	5	3

Normal Table:

					
Satisfaction	18	16	14	13	6
Capacity	12	10	9	8	4

For this case, we want a model that can be **reused** with different data.

Example Model:

```
1  enum DISH;  
2  int: capacity;  
3  
4  % Satisfaction of each dish  
5  array[DISH] of int: satisfaction;  
6  % Size of each dish  
7  array[DISH] of int: dishsize;  
8  % how many of each dish  
9  array[DISH] of var int: dishnum;  
10  
11 % Ensure that every dish served  
12 constraint forall(i in DISH)(dishnum[i] >= 0);  
13 % Ensure that the capacity does not exceed  
14 constraint sum(i in DISH)(dishsize[i] * dishnum[i]) <= capacity;  
15  
16 solve maximize sum(i in DISH)(satisfaction[i] * dishnum[i]);  
17  
18 output ["Amount = ", show(dishnum), "\n"];
```

```
1 enum DISH;  
2 DISH = {TOMYUM, BARA, DUCK};
```

- This example is [Enumerated Types](#).
- This feature allows you to treat the choice of dishes as parameters to the model.

Enumerated Types

Enumerated types, which we shall refer to as enums, are declared with a declaration of the form

```
enum <varname>;
```

An enumerated type is defined by an assignment of the form

```
<var - name> = {<item1>, ..., <itemn>};
```

where *item₁*, ..., *item_n* are the elements of the enumerated type, with name *varname*. Each of the elements of the enumerated type is also effectively declared by this definition as a new constant of that type. The declaration and definition can be combined into one line as usual.

`array[r]` of `ty`: possibly multidimensional array of elements of type `ty` (anything except array). The range expression `r` is an integer interval `l..u` in each dimension.

- Range Expression can be:
 - ▶ `l..u`, where `l` and `u` are integers or
 - ▶ enumerated type

Arrays (cont.)



```
1 % Satisfaction of each dish
2 array[DISH] of int: satisfaction;
3 satisfaction = [29,19,8];
```

- This example shows how an array is declared.
- The index set of the array `satisfaction` is `DISH`.
- Enumerated types in MiniZinc are treated similar to integers so at present the only guarantee is that only $1, 2, \dots, |\text{DISH}|$ are valid indices into the array.
- The [array access \(array lookup\)](#) `satisfaction[i]` gives the satisfaction for `DISH i`
- MiniZinc provides one- and multi-dimensional arrays.
- Elements of an array can be [anything](#) **but** another array.
- The declaration and definition can be combined into one line as usual.

- 1D arrays can be initialised using a list

```
1 array[DISH] of int: satisfaction;  
2 satisfaction = [29,19,8];
```

- 2D arrays can be initialised 2D array syntax

- ▶ start with [|
- ▶ |for separates rows
- ▶ end with |]

```
1 array[1..4, 1..3] of float: numbers1;  
2 numbers1 = [|29.5,19.3,8.2  
3             |10.0,20.35,25.2  
4             |10.3,15.4,8.6  
5             |2.3, 5.3, 1.2|];  
6  
7 array[1..4, 1..3] of float: numbers2;  
8 numbers2 = array2d(1..4, 1..3,  
9                 [29.5,19.3,8.2  
10                10.0,20.35,25.2  
11                10.3,15.4,8.6  
12                2.3, 5.3, 1.2]);
```

- MiniZinc also provides **array comprehensions**.
- An array comprehension is of the form
 - ▶ `[expression|generator1,generator2,...]`
 - ▶ `[expression|generator1,generator2,... where test]`
- Example:

```
1 [x * 3 | x in 1..8] = [3,6,9,12,15,18,21,24];  
2 [x + 2*y | x in 1..3, y in 1..4] = [3,5,7,9,4,6,8,10,5,7,9,11];  
3 [x * y | x, y in 1..4 where x < y] = [2,3,6,8,12];
```

- MiniZinc provides many built-in functions for calculating a list and a set.
 - ▶ For lists of numbers: `sum`, `product`, `max`, `min`
 - ▶ For lists of constraints: `forall`, `exists`
 - ▶ More functions can be found [here](#).
- MiniZinc provides special syntax for calls to these function and other generator functions. For example,

```
1 forall (i,j in 1..5 where i < j) (a[i] != a[j])
```

this is equivalent to

```
1 forall ([a[i] != a[j] | i,j in 1..5 where i < j])
```

Let consider the following statement:

```
forall (i,j in 1..3 where i < j) (a[i] != a[j])
```

- a is an array of numbers with index set 1..3.
- This constrains the elements in a to be **pairwise different**.
- The list comprehension evaluates to [a[1] != a[2], a[1] != a[3], a[2] != a[3]]
- Then the **forall** function returns the logical conjunction
 $a[1] \neq a[2] \wedge a[1] \neq a[3] \wedge a[2] \neq a[3]$.

Note the **exists** function returns the logical disjunction of the constraints.

```
1 % Ensure that every dish served
2 constraint forall(i in DISH)(dishnum[i] >= 0);
3 % Ensure that the capacity does not exceed
4 constraint sum(i in DISH)(dishsize[i] * dishnum[i]) <= capacity;
```

- For this example, the statements in constraint are called **Generator Expression**.
- `forall(i in range)(bool-expression)` means "for all `i` in range, `bool-expression` is true".
- `sum(i in range)(expression)` means "sum over expression for all `i` in range".

Complex Models: Data file for Banquet problem



```
1 DISH = {TOMYUM, BARA, DUCK};  
2 capacity = 18;  
3 satisfaction = [29,19,8];  
4 dishsize = [8,5,3];
```

This is an example of data file for Banquet problem.

The output will be:

```
1 Amount = [1, 1, 1] %Solution  
2 ----- %Solution found  
3 ===== %Optimal proved
```

set of ty: set of elements of type **ty**, which can be int, bool, float, or string

```
1 set of int: N = 1..5;  
2 set of bool: B;  
3 B = {true, false, true, true};
```

- This example shows how a set is declared.
- The standard set operations are provided:
 - ▶ element membership (**in**),
 - ▶ (non-strict) subset relationship (**subset**),
 - ▶ (non-strict) superset relationship (**superset**),
 - ▶ union (**union**),
 - ▶ intersection (**intersect**),
 - ▶ set difference (**diff**),
 - ▶ symmetric set difference (**symdiff**) and
 - ▶ the number of elements in the set (**card**).

- MiniZinc also provides [set comprehensions](#).
- A set comprehension is of the form
 - ▶ $\{ \text{expression} \mid \text{generator1}, \text{generator2}, \dots \}$
 - ▶ $\{ \text{expression} \mid \text{generator1}, \text{generator2}, \dots \text{ where } \text{test} \}$
- Example:

```
1 {x * 3 | x in 1..8} = {3,6,9,12,15,18,21,24};
2 {x + 2*y | x in 1..3, y in 1..4} = {3,5,7,9,4,6,8,10,5,7,9,11};
3 {x * y | x, y in 1..4 where x < y} = {2,3,6,8,12};
```

- A **global constraint** is a constraint that captures a relation between a non-fixed number of variables [6].
- MiniZinc provides a library of global constraints.
- Let consider the *cryptarithmic problem*. This problem is a type of constraint satisfaction problem.
 - ▶ This problem is about digits and its unique replacement either with alphabets or other symbols.
 - ▶ The digits (0-9) get substituted by some possible alphabets or symbols.
 - ▶ The task is to substitute each digit with an alphabet to get the result correctly.
- The constraints on cryptarithmic problem are as follows:
 - ▶ There should be a unique digit to be replaced with a unique alphabet.
 - ▶ The result should satisfy the predefined arithmetic rules, i.e., $2+2=4$, nothing else.
 - ▶ Digits should be from 0-9 only.

Global Constraints (cont.)



Let consider the following **SEND + MORE = MONEY** problem.

```
1 var 1..9: S;
2 var 0..9: E;
3 var 0..9: N;
4 var 0..9: D;
5 var 1..9: M;
6 var 0..9: O;
7 var 0..9: R;
8 var 0..9: Y;
9
10 constraint 1000 * S + 100 * E + 10 * N + D
11           + 1000 * M + 100 * O + 10 * R + E
12           = 10000 * M + 1000 * O + 100 * N + 10 * E + Y;
13
14 % All letters need to be different.
15 constraint S != E;
16 constraint S != N;
17 constraint S != D;
18 constraint S != M;
19 constraint S != O;
20 constraint S != R;
21 ....
22
23 solve satisfy;
24
25 output ["  \ (S)\(E)\(N)\(D)\n",
26         "+  \ (M)\(O)\(R)\(E)\n",
27         "=  \ (M)\(O)\(N)\(E)\(Y)\n"];
```

Global Constraints (cont.)



```
1 % All letters need to be different.  
2 constraint S != E;  
3 constraint S != N;  
4 constraint S != D;  
5 constraint S != M;  
6 constraint S != O;  
7 constraint S != R;  
8 .....
```

From these constraints, they are **tedious and long**. For this case, the global constraints will benefit.

Global Constraints (cont.)



Let consider a new model for **SEND + MORE = MONEY** problem.

```
1 include "alldifferent.mzn";
2
3 var 1..9: S;
4 var 0..9: E;
5 var 0..9: N;
6 var 0..9: D;
7 var 1..9: M;
8 var 0..9: O;
9 var 0..9: R;
10 var 0..9: Y;
11
12 constraint 1000 * S + 100 * E + 10 * N + D
13           + 1000 * M + 100 * O + 10 * R + E
14           = 10000 * M + 1000 * O + 100 * N + 10 * E + Y;
15
16 % All letters need to be different.
17 constraint alldifferent([S,E,N,D,M,O,R,Y]);
18
19 solve satisfy;
20
21 output ["  \ (S)\(E)\(N)\(D)\n",
22         "+  \ (M)\(O)\(R)\(E)\n",
23         "=  \ (M)\(O)\(N)\(E)\(Y)\n"];
```

Global Constraints (cont.)



Let consider a new model for **SEND + MORE = MONEY** problem.

```
1 % All letters need to be different.  
2 constraint alldifferent([S,E,N,D,M,O,R,Y]);
```

The statement above is used to ensure that each letter takes a different digit value. You can use this global constraint by including the following in the model.

```
1 include "alldifferent.mzn";
```

or you can use the following item to include all global constraints

```
1 include "globals.mzn";
```

For more global constraints, you can find them [here](#).

Global Constraints (cont.)



- Global constraints are constraints that may be needed in many problems
- Global constraints make *models smaller* and *solving easier*.

- `include "file-name";`
- This statement is used to
 - ▶ include the file into a MiniZinc model.
 - ▶ load library definitions of global constraints
 - ▶ break large model into small pieces

- Boolean: not, \wedge , \vee , \leftrightarrow , \rightarrow , \leftarrow , xor, forall, exists, xorall, iffall, clause, bool2int
- Integer and Float: +, -, *, div, mod, abs, pow, min, max, sum, product, =, <, <=, =>, >, !=
- Sets: union, intersect, diff, symdiff, card, in, subset, superset, set2array, array_union, array_intersect
- Strings: ++, concat, join
- Arrays: length, index_set, index_set_1of2, index_set_2of2, ..., index_set_6of6, array1d, array2d, ..., array6d

More complex constraints



- The core of MiniZinc model is **constraint**.
- In MiniZinc, a constraint is allowed to be any Boolean expression.

Let consider the following example model of Task Sequencing problem

```
1 int: n;
2 %Start time
3 array[1..n] of var int: start;
4 %Release time
5 array[1..n] of int: r;
6 %Deadline
7 array[1..n] of int: d;
8 %Processing time
9 array[1..n] of int: t;
10
11 %Dealing with release date and deadline
12 constraint forall(i in 1..n)(r[i]<=start[i] /\ start[i] + t[i] <= d[i]);
13 %Tasks cannot be overlap
14 constraint forall(i,j in 1..n where i != j)((start[i] + t[i] <= start[j]) \/ (start[j] +
15     t[j] <= start[i]));
16
17 solve satisfy;
18 output [show(start)];
```


More complex constraints (cont.)



Let consider this part of the model of Task Sequencing problem

```
1 %Tasks cannot be overlap
2 constraint forall(i,j in 1..n where i != j)((start[i] + t[i] <= start[j]) \/\ (start[j] +
3   t[j] <= start[i]));
```

- This statement ensure that two tasks cannot overlap in time.
- The \vee can be in the constraint to connect 2 small constraints.

Booleans

Boolean expressions in MiniZinc can be written using a standard mathematical syntax. The Boolean literals are `true` and `false` and the Boolean operators are conjunction, i.e. and (\wedge), disjunction, i.e. or (\vee), only-if ($\leftarrow -$), implies ($- \rightarrow$), if-and-only-if ($\leftarrow - \rightarrow$) and negation (`not`). Booleans can be automatically coerced to integers, but to make this coercion explicit the built-in function `bool2int` can be used: it returns 1 if its argument is true and 0 otherwise.

More complex constraints (cont.)



Let try to create the constraint model for The traveling salesperson problem (TSP).

Hint: You may need to use Global constraint *circuit*.

- MiniZinc provides a large set of built-in functions and predicates.
- Predicates in MiniZinc allows us to handle complex constraints of our model in a succinct way. (e.g., `alldifferent`)
- One of the most powerful modelling features of MiniZinc is to **allow modeller to define their own high-level predicates and functions.**

The predicates can be declared as follow:

```
1 predicate <pred-name> (<arg-def>, ..., <arg-def>) = <bool-exp>
```

- <pred-name> is the name of predicate.
- <arg-def>, ..., <arg-def> are the arguments of the predicate.
- <bool-exp> is the body of predicate.

Predicates are functions returning a var `bool`.

Let consider the following Task Sequencing problem example:

```
1 int: n;
2 %Start time
3 array[1..n] of var int: start;
4 %Release time
5 array[1..n] of int: r;
6 %Deadline
7 array[1..n] of int: d;
8 %Processing time
9 array[1..n] of int: t;
10
11 predicate not_overlap(var int:s1, int:t1, var int:s2, int:t2) = s1 + t1 <= s2 \ / s2 + t2
    <= s1;
12
13 %Dealing with release date and deadline
14 constraint forall(i in 1..n)(r[i]<=start[i] /\ start[i] + t[i] <= d[i]);
15 %Tasks cannot be overlap
16 constraint forall(i,j in 1..n where i != j)(not_overlap(start[i], t[i], start[j], t[j]));
17
18 solve satisfy;
19 output [show(start)];
```

```
1 predicate not_overlap(var int:s1, int:t1, var int:s2, int:t2) = s1 + t1 <= s2 \/\ s2 + t2  
   <= s1;
```

The predicate above is to ensure that each task does not overlap.

```
1 %Tasks cannot be overlap  
2 constraint forall(i,j in 1..n where i != j)(not_overlap(start[i], t[i], start[j], t[j]));
```


- Functions are defined in MiniZinc similarly to predicates but they can return more general type (not only var bool).
- The functions can be declared as follow:

```
1 function <ret-type> : <func-name> ( <arg-def>, ..., <arg-def> ) = <exp>
```

- ▶ <ret-type> is the return type of the function.
- ▶ <func-name> is the name of the function.
- ▶ <arg-def>, ..., <arg-def> are the arguments of the function.
- ▶ <exp> is the body of the function.

- Examples:

```
1 function int: double(int: x) = 2 * x;  
2 function var bool: negative(var int: x)= x < 0;
```

Note that it is possible to use `if then else endif`, `forall`, `exists` in functions.

The advantages of using functions and predicates:

- Reusable
- Modularity
- Readable
- Better common sub-expression elimination
- Definitions can be solver-specific

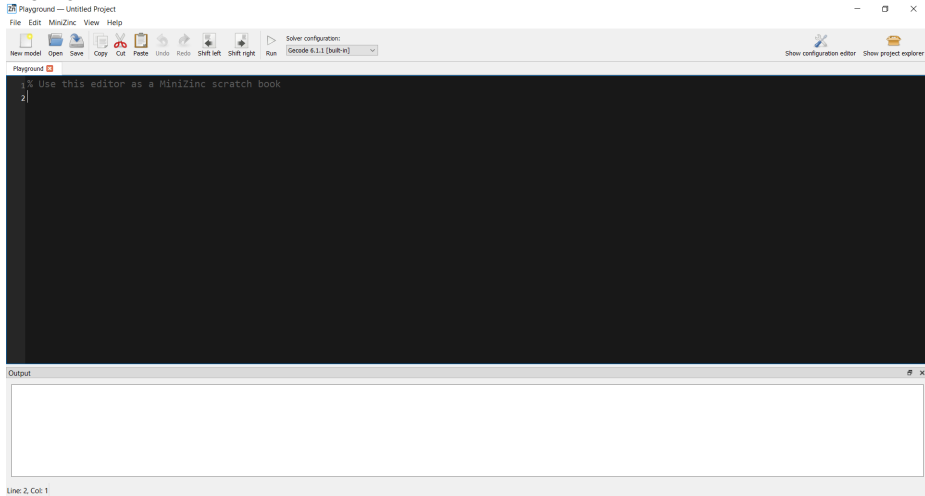
- 1 General Introduction
 - Applications
 - Sport Scheduling
 - Hong Kong International Airport (HKIA)
 - Port of Singapore
 - More Example Applications
 - CP Solving: Summary
 - CP software
- 2 MiniZinc
 - Basic Modelling in MiniZinc
 - Complex Models
 - Predicates and Functions
- 3 Other Constraint Modelling Languages
- 4 MiniZinc Software
- 5 Workshop
- 6 References

- Zinc
- EssencePrime
- Minion
- AIMMS
- AMPL
- GAMS
- SMT-LIB

- 1 General Introduction
 - Applications
 - Sport Scheduling
 - Hong Kong International Airport (HKIA)
 - Port of Singapore
 - More Example Applications
 - CP Solving: Summary
 - CP software
- 2 MiniZinc
 - Basic Modelling in MiniZinc
 - Complex Models
 - Predicates and Functions
- 3 Other Constraint Modelling Languages
- 4 **MiniZinc Software**
- 5 Workshop
- 6 References

- In order to get MiniZinc, you can download it from [here](#).
- This package includes [MiniZinc compiler](#), [MiniZinc IDE](#), and built-in solvers.
 - ▶ Gecode
 - ▶ Chuffed
 - ▶ CBC solvers
 - ▶ Gurobi solver
 - ▶ the legacy G12 solvers

Demo



Playground — Untitled Project

File Edit MiniZinc View Help

New model Open Save Copy Cut Paste Undo Redo Shift left Shift right Run Solver configurations: Gecode 6.1.1 [built-in]

Playground

```
1 % Use this editor as a MiniZinc scratch book
2 |
```

Output

Line: 2, Col: 1

- 1 General Introduction
 - Applications
 - Sport Scheduling
 - Hong Kong International Airport (HKIA)
 - Port of Singapore
 - More Example Applications
 - CP Solving: Summary
 - CP software
- 2 MiniZinc
 - Basic Modelling in MiniZinc
 - Complex Models
 - Predicates and Functions
- 3 Other Constraint Modelling Languages
- 4 MiniZinc Software
- 5 **Workshop**
- 6 References

This workshop requires you to create a MiniZinc model to solve the problem of Single Dam optimization.

The problem description:

- Objective:
 - ▶ Maximizing energy generated
 - ▶ To find the optimal water release
- Requirements:
 - ▶ A model should support for any number of days
 - ▶ The water balance need to be maintained.
 - ▶ The water release need to cover 100 percent of demands
- Question: Can your model be solved by Gecode solver? and Why?

MiniZinc Cheat Sheet

- [1] Christian Schulte, Mikael Lagerkvist, and Guido Tack.
Gecode, 2006.
- [2] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack.
Minizinc: Towards a standard cp modelling language.
In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.
- [3] George L Nemhauser and Michael A Trick.
Scheduling a major college basketball conference.
Operations research, 46(1):1–8, 1998.
- [4] Martin Henz.
Scheduling a major college basketball conference—revisited.
Operations research, 49(1):163–168, 2001.

- [5] Andy Hon Wai Chun, Steve Ho Chuen Chan, Francis Ming Fai Tsang, Dennis Wai Ming Yeung, et al.
Hkia sas: A constraint-based airport stand allocation system developed with software components.
In *AAAI/IAAI*, pages 786–793, 1999.
- [6] Willem-Jan van Hove and Irit Katriel.
Global constraints.
In *Foundations of Artificial Intelligence*, volume 2, pages 169–208.
Elsevier, 2006.