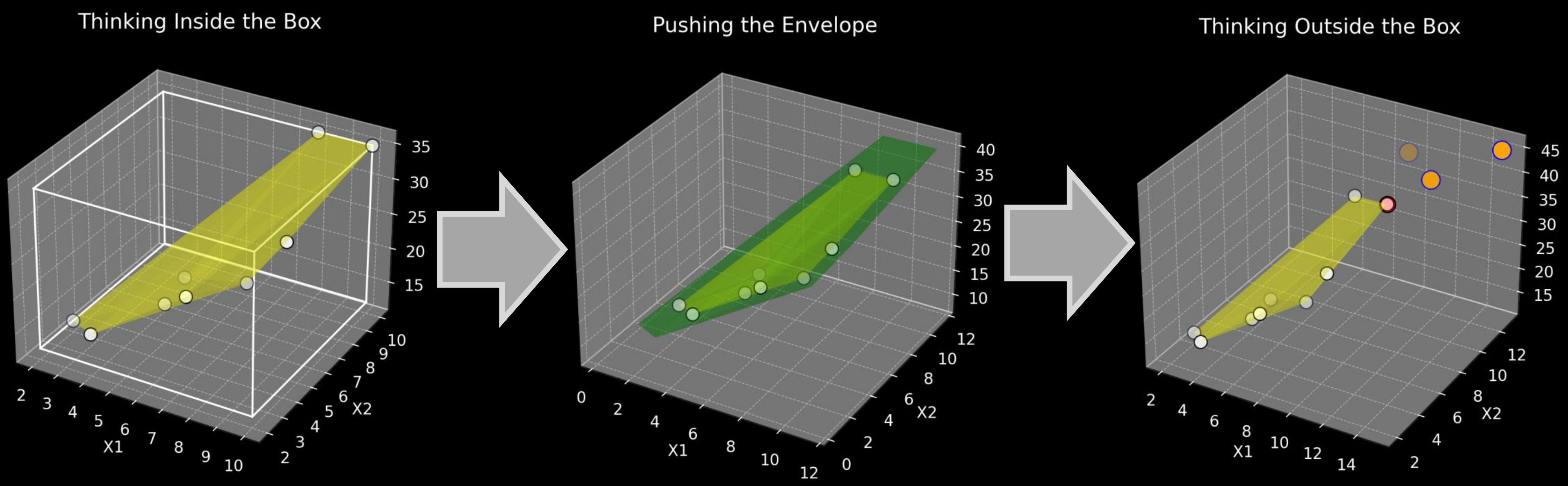
Integrating Predictive & Prescriptive Analytics in R

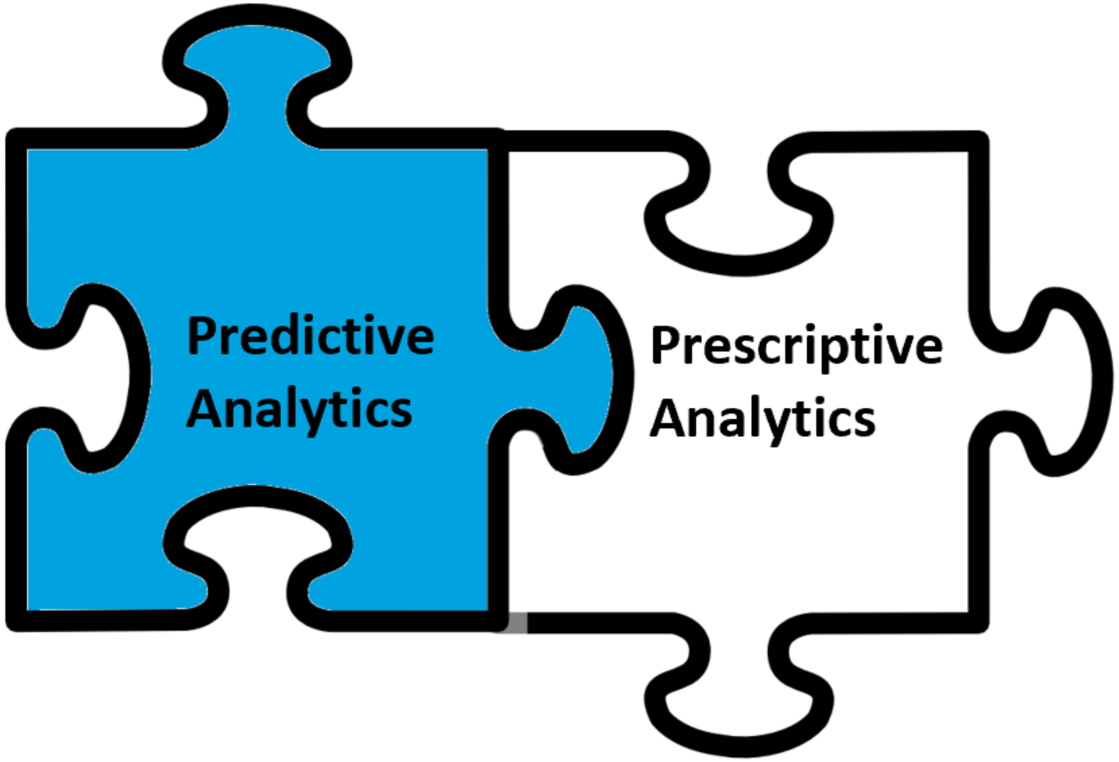
https://github.com/MatthewALanham/2025_informs/



Professor Matthew Lanham
(Former) Academic Director, MS BAIM Program
Associate Director of Student Engagements,
Krenicki Center for Business Analytics & Machine Learning
MatthewALanham.com

There has been a surge integrating prediction with optimization.

- According to Bergman, Huang et al. (2022), "*business research practice is witnessing a surge in the integration of predictive modeling and prescriptive analysis*."



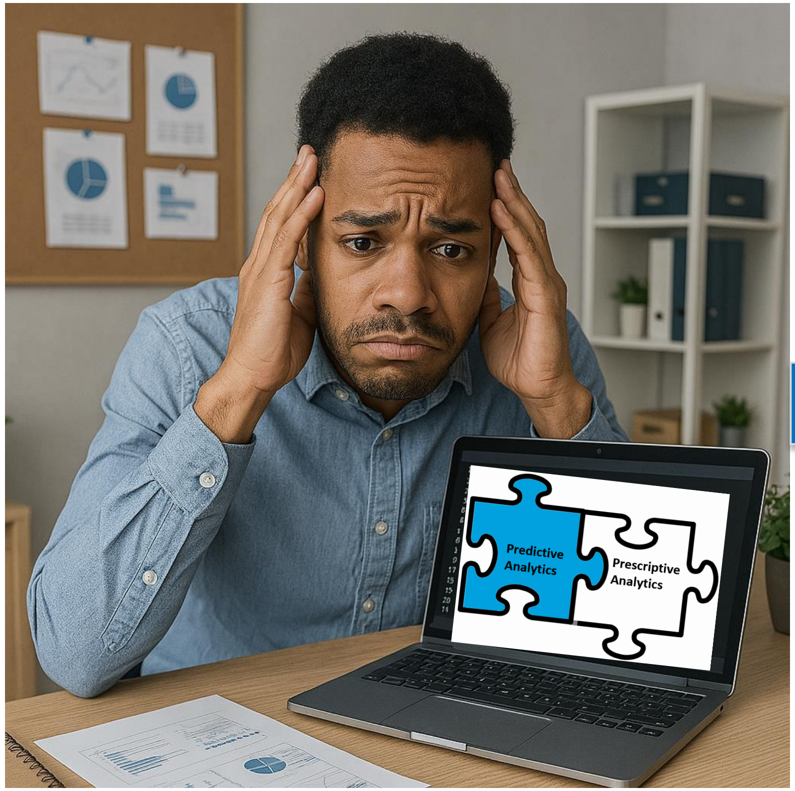
- Emergen Research (Emergen Research 2021), the global **predictive** and **prescriptive** analytics market is expected to reach $64B by 2028.
- Emergen's report forecasted growth of the analytics market in nearly every sector:
  - Retail (assortment planning)
  - Energy
  - Manufacturing (optimal process settings)
  - Automotive
  - Defense
  - Healthcare
  - Sports (dynamic ticket pricing)
  - Entertainment (vacation rentals)

Bergman, D., et al. (2022). "JANOS: an integrated predictive and prescriptive modeling framework." INFORMS Journal on computing **34**(2): 807-816.
Emergen Research (2021) Predictive and Prescriptive Analytics Market.

Today: Predictive to prescriptive modeling considerations (Using R!)

Some of you in the audience may be challenged on where to begin:

- How do I integrate a predictive model into a formal mathematical optimization model?
- Am I making sure I am <u>NOT</u> getting "risky" decision recommendations?
- There are not really a lot of examples in this area, particularly for R folks.



R

Next, connect with the experts here at INFORMS

AMPL

FICO

GAMS
MODEL – SOLVE – DEPLOY

GUROBI
OPTIMIZATION

sas

and many more!

https://github.com/MatthewALanham/2025_informs/

Two-stage design

There are many business problems where a decision-maker wants to make an “optimal” decision using analytical frameworks, methods, and models. Practitioners often use their observational data to:

1. Describe what they are seeing
2. Predict what might happen in the future
3. Then Optimize actions to take

A common modeling design is taking a **two-stage** approach:

- **Stage 1:** Predict what you want to know
- **Stage 2:** Optimize what actions you should take

$y = f(X) + \varepsilon \longrightarrow \min \hat{y} \ \ s.t. Z \longrightarrow$ Decision Quality?

https://github.com/MatthewALanham/2025_informs/

What are some ways to try it?

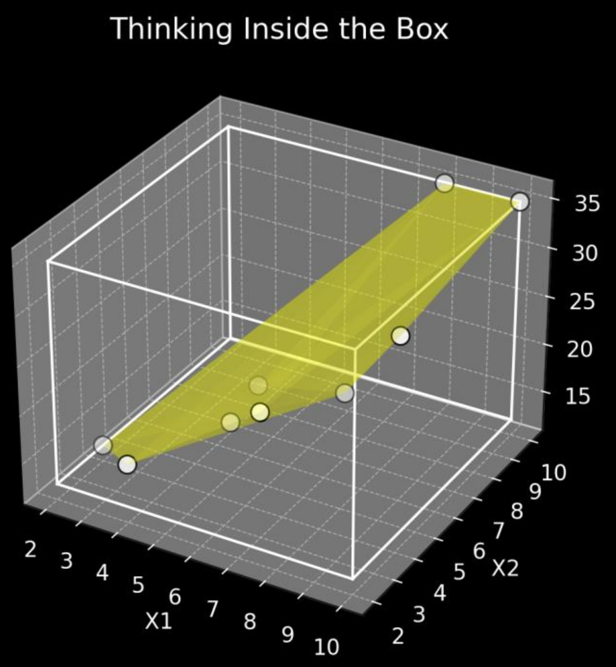
Two-stage designs are the most common way to interface the prediction with the optimization. However, according to *Bertsimas and Kallus (2020)*, "*it is not clear how to go from a good prediction to a good decision.*"

$y = f(X) + \varepsilon$ ⟶ $\min \hat{y}$ $s.t. Z$ ⟶ Decision Quality?

Lets see some practical modeling considerations….

Bertsimas, D. and N. Kallus (2020). "From predictive to prescriptive analytics." Management Science **66**(3): 1025-1044.

Thinking Inside the Box



Professor Matthew Lanham
(Former) Academic Director, MS BAIM Program
Associate Director of Student Engagements,
Krenicki Center for Business Analytics & Machine Learning
MatthewALanham.com

A good starting point begins with a good problem

First, lets focus on the linear regression model, which falls under the general umbrella of statistical learning and parametric models.

1. $\boldsymbol{y}$ is the target variable in our predictive model y: $\{y \in \mathbb{R}\}$
2. $\boldsymbol{X}$ are the input features of the predictive model; numeric x: $\{x \in \mathbb{R}\}$ or binary x: $\{x \in 0,1\}$
3. $\boldsymbol{\varepsilon}$ is the predictive model error term $\varepsilon \sim N(0, \sigma)$

Equation 1: Predictive model

$$y = f(X) + \varepsilon$$

Now consider cases where the predictive model has been estimated and the firm must identify what actions would lead to the best outcome.

Equation 2: Prescriptive (i.e., optimization) model

$$\begin{gathered} \min \hat{y} \\ s.t. Z \end{gathered}$$

Equation 2 might be to minimize our prediction (say, of cost) subject to some constraints Z.

Stage 1: estimate the relationship

Consider fitting a linear regression model having numeric features, then formulating an optimization model using those estimated parameter coefficients to identify "optimal" decision recommendations.

Example: sample carpet data
$x_1 \in\ (2,4,4,5,7,7,9,8,10)$
$x_2 \in\ (4,8,2,5,7,3,6,10,10)$
$y \in\ (12.5, 13.3, 16.9, 16.4, 17.8, 23.7, 27.8, 35, 35.1)$

If an OLS model is fit to this data, it would yield an estimated predictive model of:

$$\hat{y} = 2.674 + 2.685x_1 + 0.437x_2$$

This would create the following plane shown through three-dimensional space.

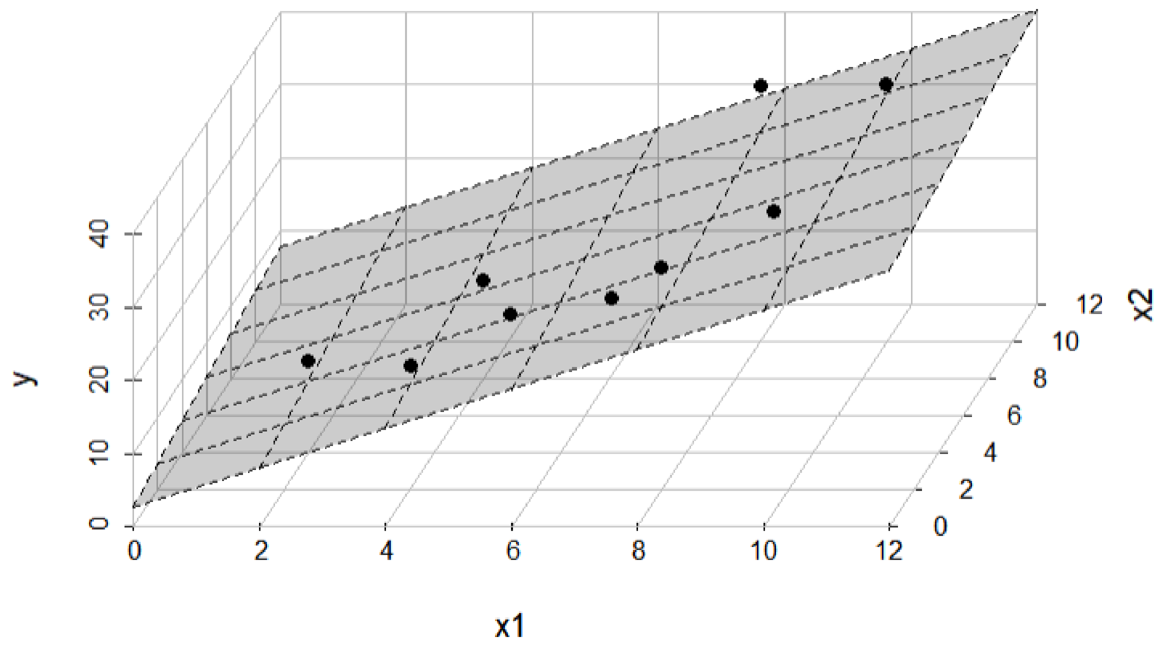


Figure 4: OLS best fit plane

https://github.com/MatthewALanham/2025_informs/

Stage 2: predictive model part of optimization model

You might then take the estimated predictive model and formulate it into an optimization model to obtain the “best” solution or decision.

Assumptions:

- Target variable y in the predictive model is a business performance measure that a decision-maker is seeking to improve (e.g., “quality”)
- You might decide to formulate the predictive model into the optimization as either:
    1) the objective function, or
    2) as a constraint

- Expected quality must be greater than or equal to 29
- Total budgetary cost is $225

Model formulation 1: Maximize quality

$$\max\ y = 2.674 + 2.685x_1 + 0.437x_2 \qquad [1]$$

s.t.

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \qquad [2]\ (\text{“quality constraint”})$$

$$\$10x_1 + \$20x_2 \leq \$225 \qquad [3]\ (\text{“budgetary constraint”})$$

$$x_i \geq 0 \qquad [4]\ (\text{“non-negativity”})$$

No constraints can lead to extrapolations

After solving the problem, we find the suggested optimal decision is:

- $x_1^* = 22.5$ and $x_2^* = 0$
- leads to an $E[y] = 63.09$ (quality estimate) and an $E[Cost] = \$225$.

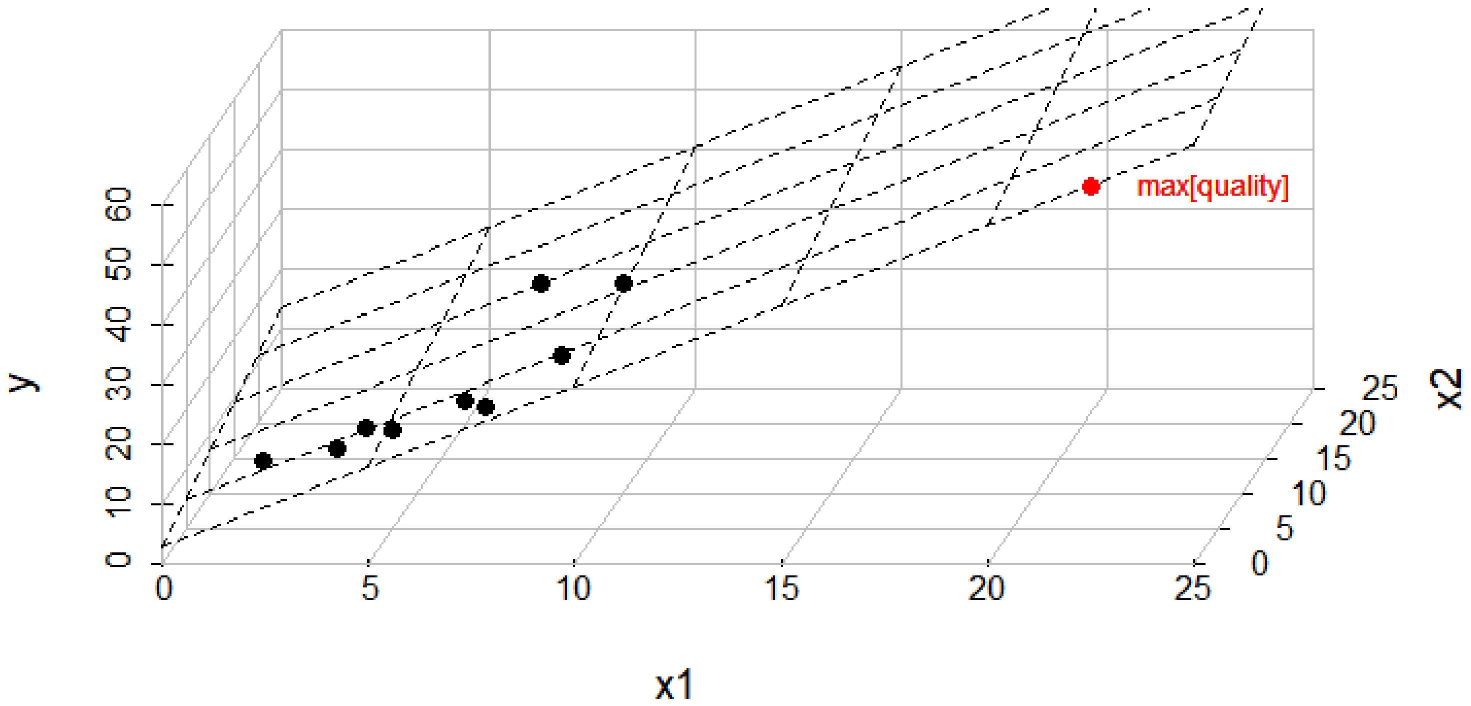


Figure 5: Suggested optimal decision for maximizing quality

When developing a predictive model to eventually support decision-making, are you thinking and designing with the end decision support in mind?

Is predictive modeling process aligned with the decision modeling process?

Say one of the variables x_2 is not statistically significant. Do you leave it in or remove it?

Example:

$\hat{y} = 2.674 + 2.685x_1 + 0.437x_2$

```
MODEL INFO:
Observations: 9
Dependent Variable: y
Type: OLS linear regression

MODEL FIT:
F(2,6) = 12.07, p = 0.01
R² = 0.80
Adj. R² = 0.73

Standard errors: OLS
```

	Est.	2.5%	97.5%	t val.	p
(Intercept)	2.67	-7.93	13.28	0.62	0.56
x1	2.69	0.88	4.49	3.63	0.01
x2	0.44	-1.21	2.09	0.65	$\boxed{0.54}$

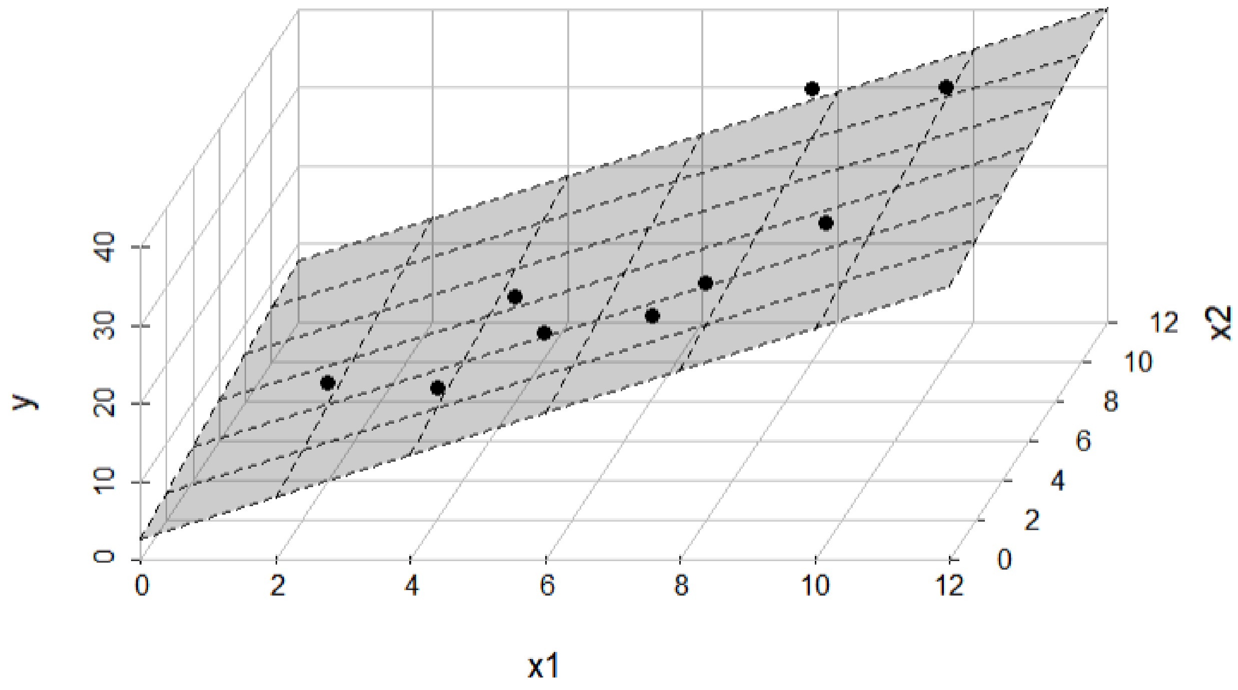


Figure: OLS best fit plane

https://github.com/MatthewALanham/2025_informs/

Keep the end in mind..

Observations:

1. Suggested decision recommendations may deviate significantly from historical data. Is that okay?
2. Severe extrapolation – don't overlook possible implicit predictive modeling assumptions.
3. What if decision variables are removed because they are statistically insignificant? Consider in this problem that x_1 and x_2 are manufacturing process settings.
4. Prediction error
    - Adjusted R^2=0.7347
    - "Expected quality" level of $E[y] = 63$

   The recommended decision has error directly from the predictive model. Are you aware and accounting for this error in stage 2?

Stage 1: Prediction

$$\hat{y} = 2.674 + .685x_1 + 0.437x_2$$

Stage 2: Optimization

$$\max\ y = 2.674 + 2.685x_1 + 0.437x_2 \quad [1]$$

s.t.

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \quad [3]$$

$$x_i \geq 0 \quad [4]$$

https://github.com/MatthewALanham/2025_informs/

Curtailing extrapolation risk using data-driven constraints

How could we make "less risky" **data-supported** decision recommendations?

Model formulation 2: Maximize quality with min-max constraints (a.k.a. "box constraints")

$$\max y = 2.674 + 2.685x_1 + 0.437x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \quad [3]$$

$$\min(x_1) \leq x_1 \leq \max(x_1) \quad [4]$$

$$\min(x_2) \leq x_2 \leq \max(x_2) \quad [5]$$

$$x_i \geq 0 \quad [6]$$

- $x_1^* = 10.0$; $x_2^* = 6.25$
- $E[y] = 32.26$ (previously $E[y] = 63.09$)
- $E[Cost] = \$225$ (same as before)

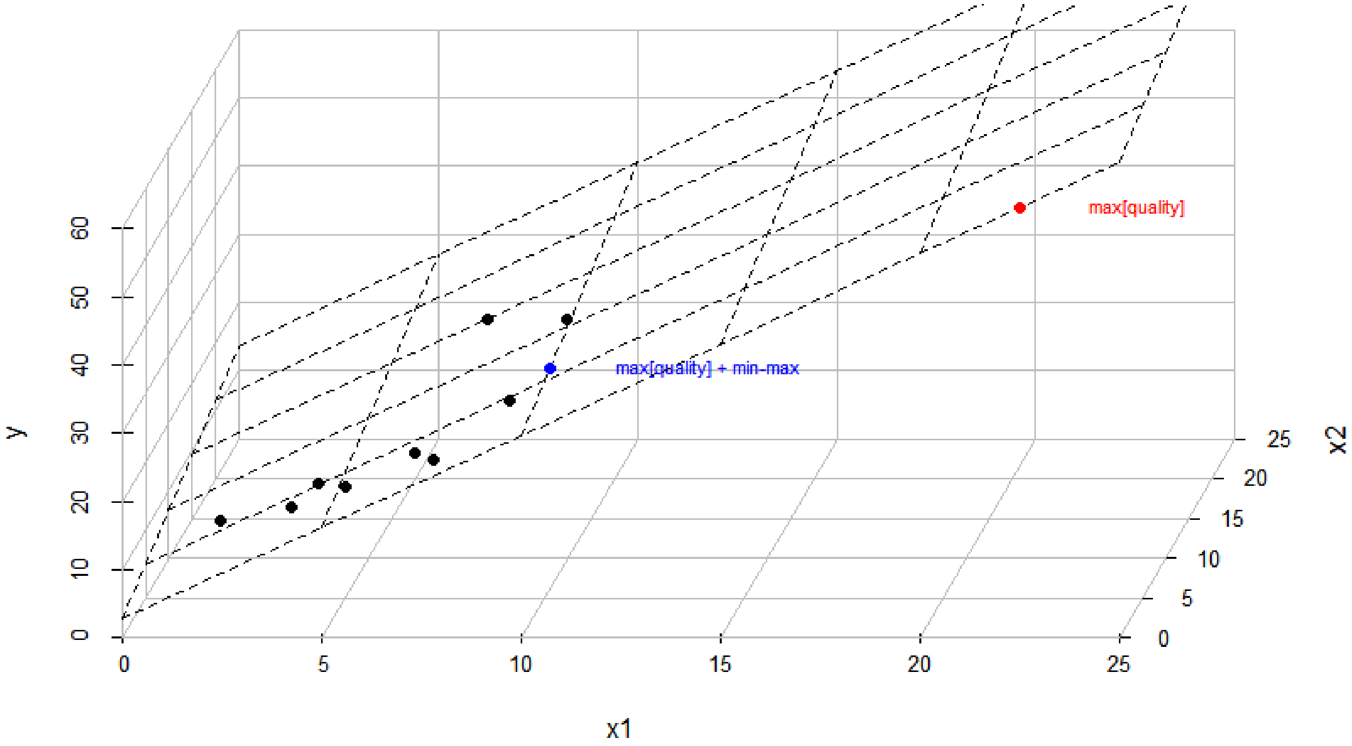


Figure 7: Suggested optimal decision for maximizing quality with min-max constraints

https://github.com/MatthewALanham/2025_informs/

Min-max constraints

In our new solution using min-max constraints, the business requirements of quality and costs are still satisfied.

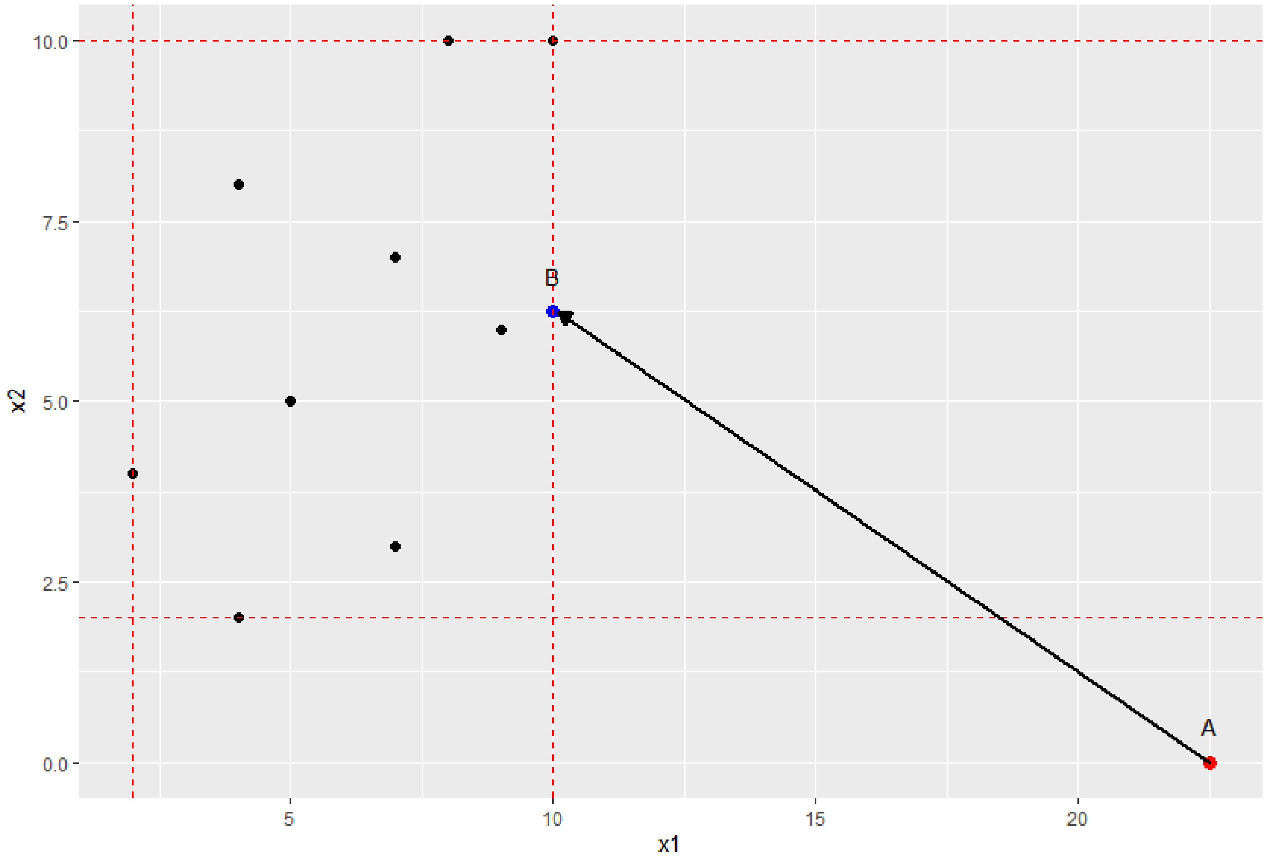


Figure 8: Suggested optimal decision change adding min-max constraints

Is just including min-max constraints for these types of problems enough?

Curtailing extrapolation risk using data-driven constraints

Consider if x_1 or x_2 boundary points (points on red dotted lines) were actually much further away from the other points – such as an “outlier.” For example (point D).

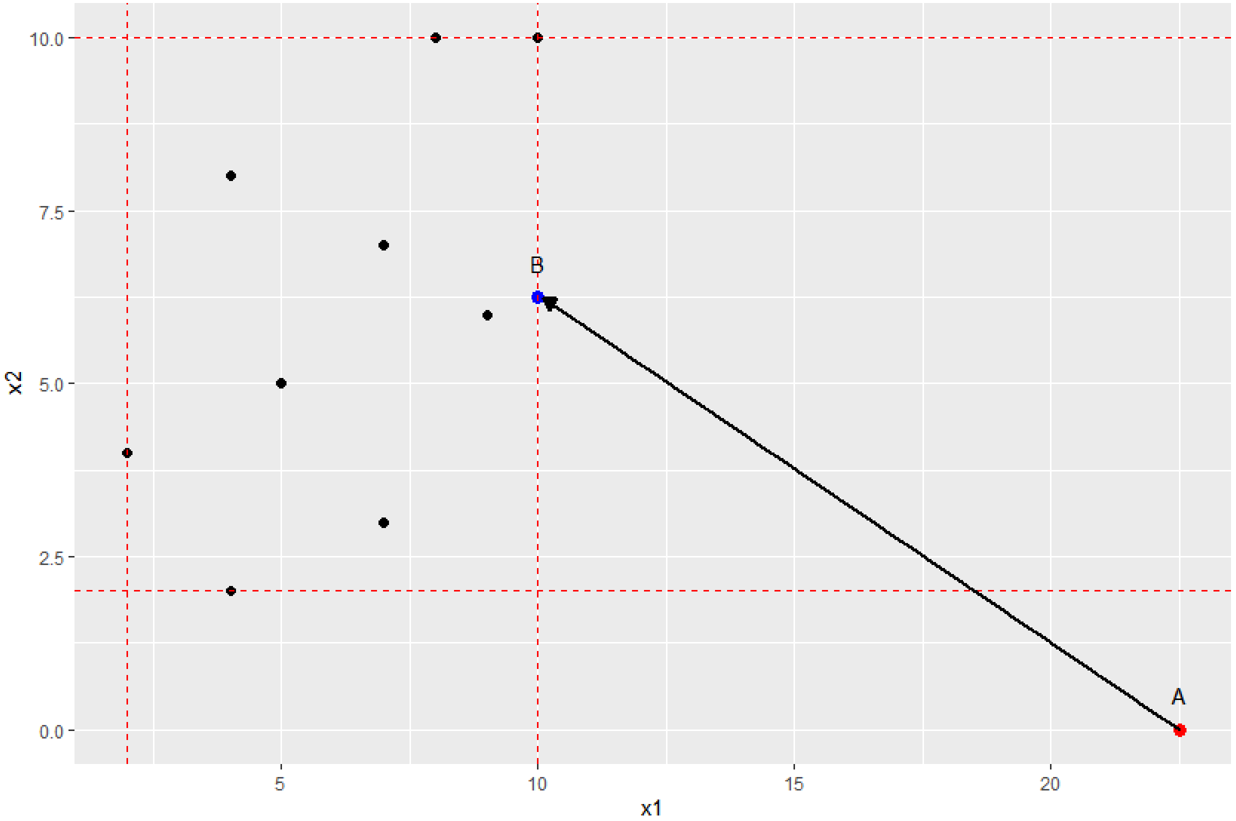


Figure 8: Suggested optimal decision change adding min-max constraints

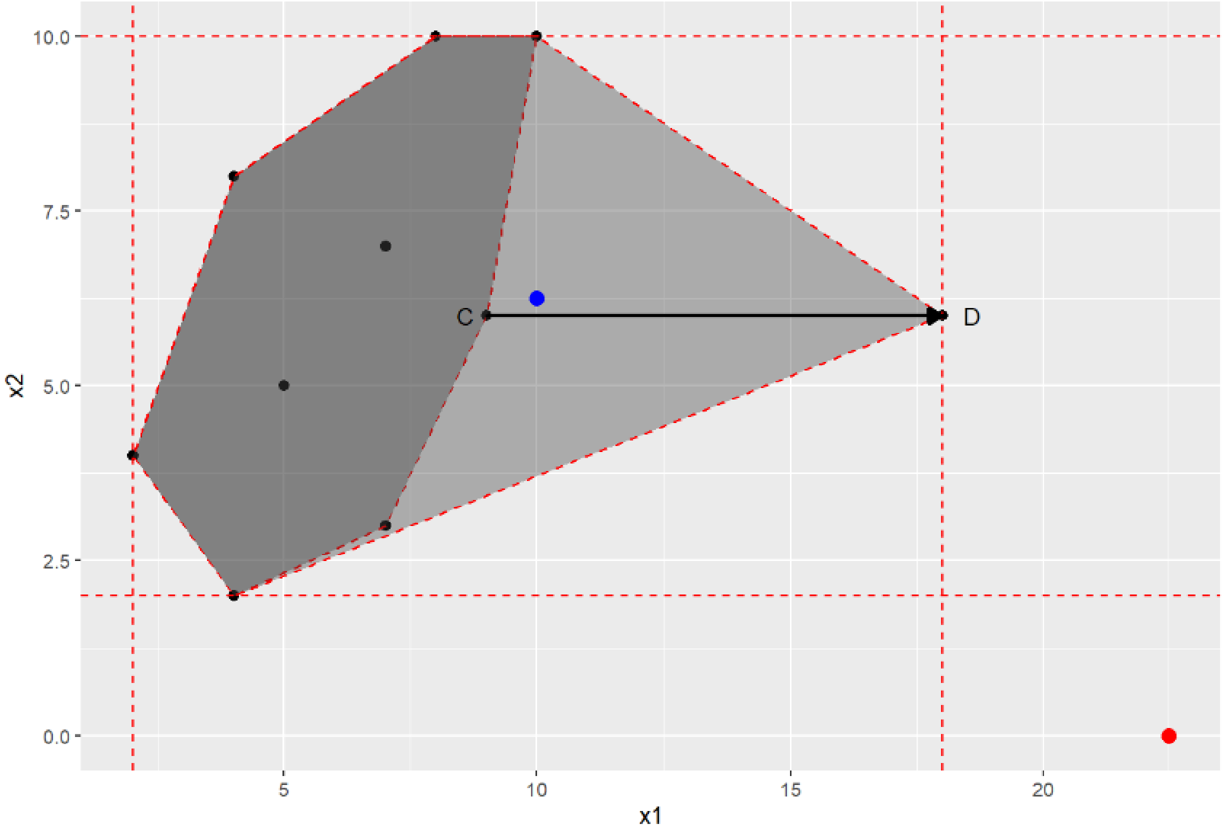


Figure 9: Min-max constraints are riskier when data is not tight or outliers exists

If you could envelop the historical data more tightly, maybe this could lead to improved “data supported” decisions?

R Time!

```r
#####################################################################################
# Predictive to Prescriptive Analytics
# Matthew A. Lanham
# Section 1. Example ideas and graphs
#####################################################################################
#####################################################################################
#              Section 1. Example ideas and graphs
#####################################################################################
# This example dataset demonstrates the idea of prediction
# and optimization to motivate future considerations
#####################################################################################
x1 <- c(2,4,4,5,7,7,9,8,10)
x2 <- c(4,8,2,5,7,3,6,10,10)
y <- c(12.5,13.3,16.9,16.4,17.8,23.7,27.8,35,35.1)
d <- data.frame(cbind(y,x1,x2))
rm(x1,x2,y)
cost_parameters = c(10,20)
cost_constraint_param = 225
quality_constraint_param = 29
```

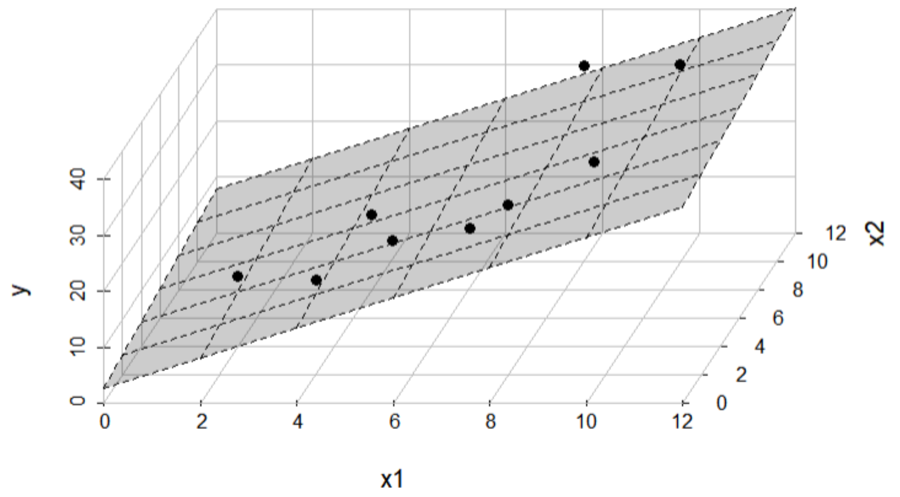
	y	x1	x2
1	12.5	2	4
2	13.3	4	8
3	16.9	4	2
4	16.4	5	5
5	17.8	7	7
6	23.7	7	3
7	27.8	9	6
8	35.0	8	10
9	35.1	10	10

https://github.com/MatthewALanham/2025_informs/

R Time!

https://github.com/MatthewALanham/2025_informs/

```r
##################################################################################
# Visualize relationship
##################################################################################
#3d visual of problem
names(d)
par(mfrow=c(1,1))
library(scatterplot3d)
plot3d <- scatterplot3d(x = d$x1, y = d$x2, z = d$y
                        , xlab="x1", ylab="x2", zlab="y"
                        , xlim=c(0,12), ylim=c(0,12), zlim=c(0,40)
                        , bg="black", color="white", angle=65, scale.y=0.7
                        , cex.symbols=1.6, pch=16
                        , box=F, col.grid="grey", lty.grid=par("lty"))
# add light grey grid
source('http://www.sthda.com/sthda/RDoc/functions/addgrids3d.r')
addgrids3d(d[,c(3,2,1)], grid=c("xy", "xz", "yz")
           , angle = 65, xlim=c(0,12), ylim=c(0,12), zlim=c(0,40)
           , scale.y=0.7)
# put points back on top of grid
plot3d$points3d(x=d$x1, y=d$x2, z=d$y, type="p", pch=16, cex=1, col="black")
# fit a multiple linear regression
f <- lm(y ~ x1 + x2, data=d)
summary(f)
# show output nicely
# https://cran.r-project.org/web/packages/jtools/vignettes/summ.html
library(jtools)
summ(f, pvals = T, confint=T)
# fit a multiple linear regression WITH interaction terms
fint <- lm(y ~ x1 + x2 + I(x1*x2), data=d)
summary(fint)
# add the fitted regression plane to the 3d-plot
plot3d$plane3d(f, lty.box="dashed", draw_lines=T, draw_polygon=T)
```



R Time!

```r
###########################################################################
# maximize quality
###########################################################################
library(lpSolveAPI)
library(lpSolve)
# we'll just start with 0 constraints and add them later
# there are two decision variables.
(lps.model <- make.lp(nrow=0, ncol=3))
set.type(lps.model, columns=1, type="real") # decision variable is "real" number
set.type(lps.model, columns=2, type="real") # decision variable is "real" number
set.type(lps.model, columns=3, type="real") # decision variable is "real" number
# set objective function
lp.control(lps.model, sense="max")
set.objfn(lps.model, obj=c(f$coefficients[["(Intercept)"]]
                           ,f$coefficients[["x1"]]
                           ,f$coefficients[["x2"]]))
# define constraints
add.constraint(lps.model, c(f$coefficients[["(Intercept)"]]
                            ,f$coefficients[["x1"]],f$coefficients[["x2"]])
               , ">=", quality_constraint_param)
add.constraint(lps.model, c(0,cost_parameters), "<=", cost_constraint_param)
add.constraint(lps.model, c(1,0,0), "=", 1) #makes first d.v. 1 (this is to keep
#intercept term in)
# (Optional - can provide some constrain names - not necessary)
dimnames(lps.model) <- list(c("quality constraint", "cost constraint"
                              ,"intercept constraint"),
                            c("intercept scalar", "# x1", "# x2"))
```

Stage 1: Prediction

$$\hat{y} = 2.674 + .685x_1 + 0.437x_2$$

```
> f$coefficients
(Intercept)          x1          x2
  2.6746044   2.6851545   0.4374529
```

Stage 2: Optimization

$$\max\ y = 2.674 + 2.685x_1 + 0.437x_2 \qquad [1]$$

s.t.

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \qquad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \qquad [3]$$

$$x_i \geq 0 \qquad [4]$$

https://github.com/MatthewALanham/2025_informs/

R Time!

```r
# lets review our LP model
lps.model
# solve the model
solve(lps.model)
get.objective(lps.model) # optimal obj. value (i.e. our maximum profit)
get.variables(lps.model) # optimal soln of d.v.'s (i.e. our decisions to make)

# save results
(results <- data.frame(matrix(NA,nrow=1,ncol=5)))
(names(results) <- c("Model","x1","x2","Exp[y]","Exp[Cost]"))
results$Model <- "Max[Quality]"
results$x1 <- get.variables(lps.model)[[2]]
results$x2 <- get.variables(lps.model)[[3]]
results$"Exp[y]" <- get.objective(lps.model)
results$"Exp[Cost]" <- sum(c(0,cost_parameters)*get.variables(lps.model))
results
```

Stage 2: Optimization

$$\max\ y = 2.674 + 2.685x_1 + 0.437x_2 \qquad [1]$$

s.t.

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \qquad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \qquad [3]$$

$$x_i \geq 0 \qquad [4]$$

```
> get.objective(lps.model)
[1] 63.09058
> get.variables(lps.model)
[1]  1.0 22.5  0.0
```

```
> lps.model
Model name:
                          intercept scalar            # x1            # x2
Maximize                  2.67460437076     2.6851544838  0.437452901281
quality constraint        2.67460437076     2.6851544838  0.437452901281  >=   29
cost constraint                       0               10              20  <=  225
intercept constraint                  1                0               0   =    1
Kind                                Std              Std             Std
Type                               Real             Real            Real
Upper                               Inf              Inf             Inf
Lower                                 0                0               0
```

https://github.com/MatthewALanham/2025_informs/

R Time!

```r
##########################################################################
# maximize quality w/ min-max constraints
##########################################################################
library(lpSolveAPI)
library(lpSolve)
# we'll just start with 0 constraints and add them later
# there are two decision variables.
(lps.model <- make.lp(nrow=0, ncol=3))
set.type(lps.model, columns=1, type="real") # decision variable is "real" number
set.type(lps.model, columns=2, type="real") # decision variable is "real" number
set.type(lps.model, columns=3, type="real") # decision variable is "real" number
get.type(lps.model) # to see what types are defined for each D.V.
# set objective function
lp.control(lps.model, sense="max")
set.objfn(lps.model, obj=c(f$coefficients[["(Intercept)"]]
                           ,f$coefficients[["x1"]],f$coefficients[["x2"]]))
# define constraints
add.constraint(lps.model, c(f$coefficients[["(Intercept)"]]
                            ,f$coefficients[["x1"]],f$coefficients[["x2"]])
               , ">=", quality_constraint_param)
add.constraint(lps.model, c(0,cost_parameters), "<=", cost_constraint_param)
add.constraint(lps.model, c(1,0,0), "=", 1) #makes first d.v. 1 (this is to
#keep intercept term in)
add.constraint(lps.model, c(0,1,0), "<=", max(d$x1)) #min-max for x1
add.constraint(lps.model, c(0,1,0), ">=", min(d$x1)) #min-max for x1
add.constraint(lps.model, c(0,0,1), "<=", max(d$x2)) #min-max for x2
add.constraint(lps.model, c(0,0,1), ">=", min(d$x2)) #min-max for x2
# in the lpSolve linear program model object.
dimnames(lps.model) <- list(c("quality constraint", "cost constraint"
                              ,"intercept constraint","min x1 constr."
                              ,"max x1 constr.","min x2 constr.","max x2 constr."),
                            c("intercept scalar", "# x1", "# x2"))
```

Stage 1: Prediction

$$\hat{y} = 2.674 + .685x_1 + 0.437x_2$$

```
> f$coefficients
(Intercept)          x1          x2
  2.6746044   2.6851545   0.4374529
```

Stage 2: Optimization

$$\max y = 2.674 + 2.685x_1 + 0.437x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \quad [3]$$

$$\min(x_1) \leq x_1 \leq \max(x_1) \quad [4]$$

$$\min(x_2) \leq x_2 \leq \max(x_2) \quad [5]$$

$$x_i \geq 0 \quad [6]$$

https://github.com/MatthewALanham/2025_informs/

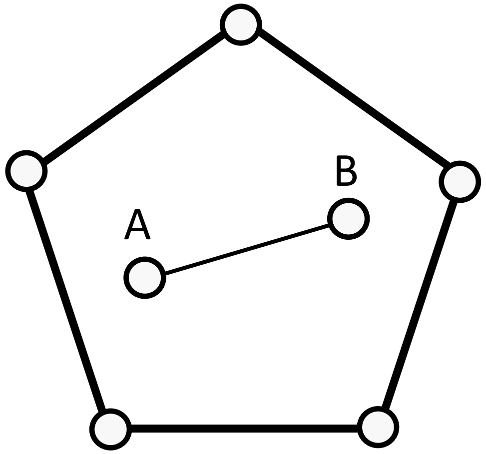
Can we bound the data tighter?

(convex hull constraints)

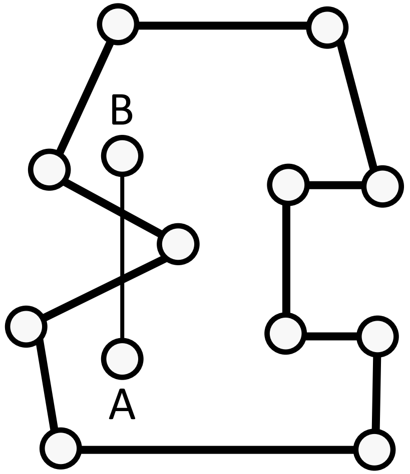
Professor Matthew Lanham
(Former) Academic Director, MS BAIM Program
Associate Director of Student Engagements,
Krenicki Center for Business Analytics & Machine Learning
MatthewALanham.com

Convex Hull Constraints

A **convex set** or convex region is a subset that intersects every line into an individual line segment. Formally, a subset C of S is convex if, for all x and y in C, the line segment connecting x and y is included in C.



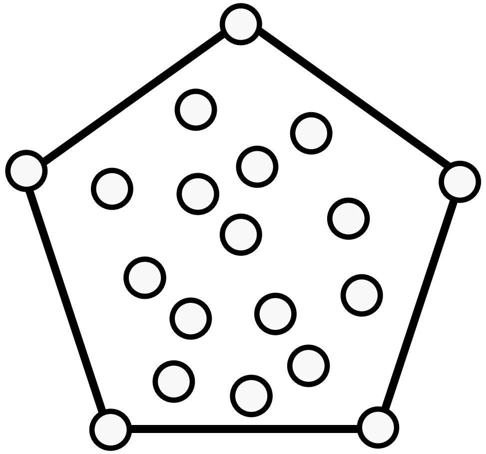
Convex set



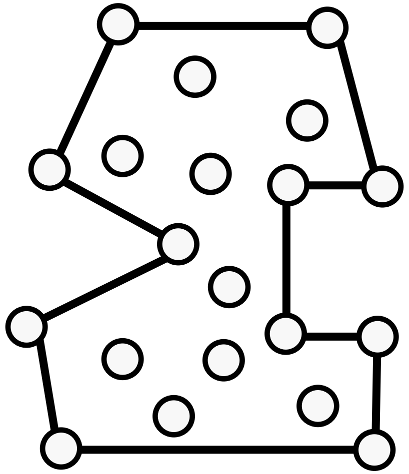
Non-convex set

The **convex hull** is the set of all convex combinations of points in a convex set. Visually, it is the bounded or enclosed area of a subset. Formally, the convex hull of X is the intersection of all convex sets Y that contain X.

$$\operatorname{conv}(X) := \bigcap \{Y \subseteq A : X \subseteq Y, Y \text{ convex}\}$$



Convex hull



Concave hull

Here is how you can create convex hull constraints

Algorithm:

1. Identify the vertices of the hull denoted by the set of points $(\boldsymbol{v_{j1}}, \boldsymbol{v_{j2}})$ for all $j \in$ J.
2. Non-negative decision variables' are included in the model as $\boldsymbol{w_j}$, where j is the index of the jth vertex.
3. Sum of the vertex weights w_j must sum to 1.

Model formulation 3: Maximize quality with convex hull constraints

$$\max y = \; 2.674 + 2.685x_1 + 0.437x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$
$$\$10x_1 + \$20x_2 \leq \$225 \quad [3]$$
$$\sum_{i=1}^{J} w_j v_{j1} = x_1 \quad [4]$$
$$\sum_{i=1}^{J} w_j v_{j2} = x_2 \quad [5]$$
$$\sum_{i=1}^{J} w_j = 1 \quad [6]$$
$$w_j, x_i \geq 0 \quad [7]$$

Note: Every problem you encounter will likely have a different # of vertices.

Together constraints [4], [5], and [6] ensure that any linear combination of points within the hull are considered "data supported" and points outside the hull are not considered as possible decision recommendations.

Data-driven constraints via convex hull constraints

Model formulation 3: Maximize quality with convex hull constraints

$$\max y = \ 2.674 + 2.685x_1 + 0.437x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \quad [3]$$

$$\sum_{i=1}^{J} w_j v_{j1} = x_1 \quad [4]$$

$$\sum_{i=1}^{J} w_j v_{j2} = x_2 \quad [5]$$

$$\sum_{i=1}^{J} w_j = 1 \quad [6]$$

$$w_j, x_i \geq 0 \quad [7]$$

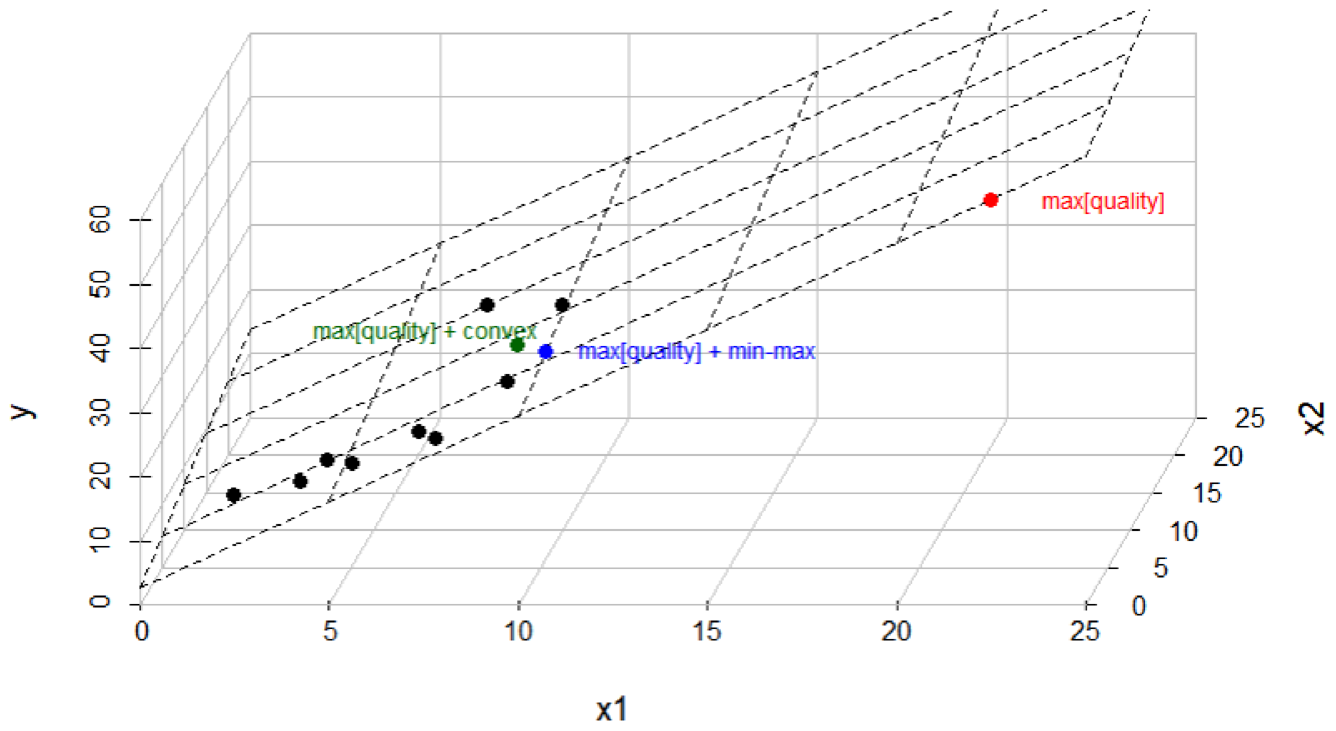


Figure 10: Decision recommendation for quality based on three model formulations

- $x_1^* = 9.16$; $x_2^* = 6.66$
- $E[y] = 30.20$ (previously $E[y] = 32.26$)
- $E[Cost] = \$225$ (same as before)

https://github.com/MatthewALanham/2025_informs/

Data-driven constraints via convex hull constraints

The new point C solution provides an expected quality that is not extrapolated like the first model formulation (point A solution), and also falls within the convex hull containing the historical values, which the second model formulation did not (point B solution)

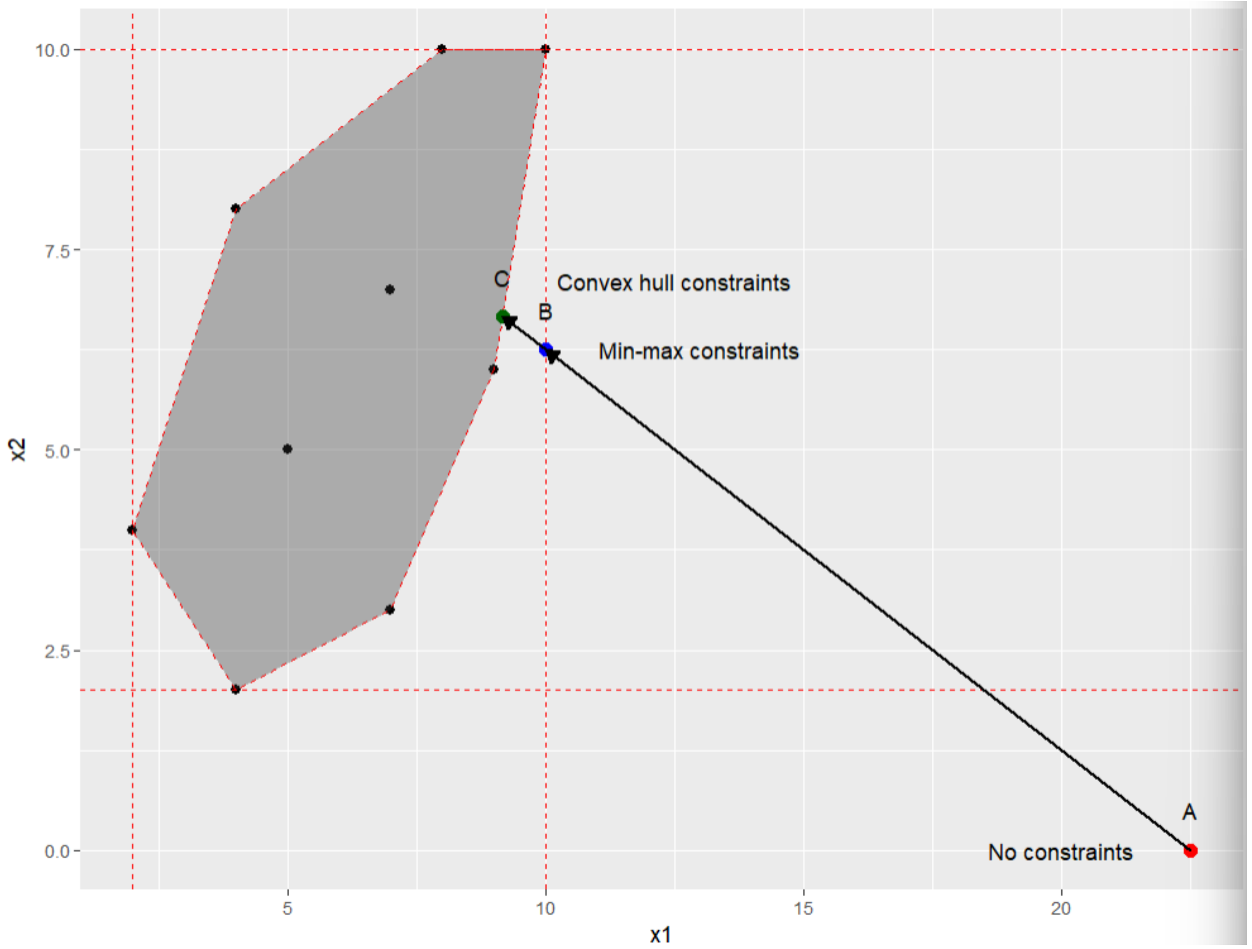


Figure 11: Outcomes using no historical data constraints, box constraints, and convex hull constraints.

R Time!

https://github.com/MatthewALanham/2025_informs/

```
####################################
# Find vertices of convex hull
####################################
library(tidyverse)
(hull <- chull(d))
(hull <- d %>%
    slice(chull(x1, x2)))
(hull <- hull[,2:3])
(num_vertex_constr <- nrow(hull))
```

1. We find the convex hull vertices here.
2. Next we will incorporate those into the optimization model as constraints.

$$\sum_{i=1}^{J} w_j v_{j1} = x_1 \qquad [4]$$

$$\sum_{i=1}^{J} w_j v_{j2} = x_2 \qquad [5]$$

$$\sum_{i=1}^{J} w_j = 1 \qquad [6]$$

https://github.com/MatthewALanham/2025_informs/

R Time!

https://github.com/MatthewALanham/2025_informs/

```r
################################################################################
# max quality + convex-hull constraints
################################################################################
# there are two decision variables. Make them "real"
(lps.model <- make.lp(nrow=0, ncol=3+num_vertex_constr))
for (i in 1:(3+num_vertex_constr)){
  set.type(lps.model, columns=i, type="real")
}
get.type(lps.model) # to see what types are defined for each D.V.
# set objective function
lp.control(lps.model, sense="max")
set.objfn(lps.model, obj=c(f$coefficients[["(Intercept)"]]
                           ,f$coefficients[["x1"]],f$coefficients[["x2"]]
                           ,rep(0,num_vertex_constr)))
# define constraints
add.constraint(lps.model, c(f$coefficients[["(Intercept)"]]
                            ,f$coefficients[["x1"]],f$coefficients[["x2"]]
                            ,rep(0,num_vertex_constr))
               , ">=", quality_constraint_param)
add.constraint(lps.model, c(0,cost_parameters,rep(0,num_vertex_constr))
               , "<=", cost_constraint_param)
# makes first d.v. 1 (this is to keep intercept term in)
add.constraint(lps.model, c(1,0,0,rep(0,num_vertex_constr)), "=", 1)
add.constraint(lps.model, c(0,1,0,rep(0,num_vertex_constr)), "<=", max(d$x1)) #mi
add.constraint(lps.model, c(0,1,0,rep(0,num_vertex_constr)), ">=", min(d$x1)) #mi
add.constraint(lps.model, c(0,0,1,rep(0,num_vertex_constr)), "<=", max(d$x2)) #mi
add.constraint(lps.model, c(0,0,1,rep(0,num_vertex_constr)), ">=", min(d$x2)) #mi
add.constraint(lps.model, c(0,-1,0,hull[,1]), "=", 0) #convex hull x1s = dv1
add.constraint(lps.model, c(0,0,-1,hull[,2]), "=", 0) #convex hull x2s = dv2
add.constraint(lps.model, c(0,0,0,rep(1,num_vertex_constr)), "=", 1) #sum of weig
```

Stage 1: Prediction

$$\hat{y} = 2.674 + .685x_1 + 0.437x_2$$

```
> f$coefficients
(Intercept)          x1          x2
  2.6746044   2.6851545   0.4374529
```

Stage 2: Optimization

$$\max y = \quad 2.674 + 2.685x_1 + 0.437x_2 \qquad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \qquad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \qquad [3]$$

$$\sum_{i=1}^{J} w_j v_{j1} = x_1 \qquad [4]$$

$$\sum_{i=1}^{J} w_j v_{j2} = x_2 \qquad [5]$$

$$\sum_{i=1}^{J} w_j = 1 \qquad [6]$$

$$w_j, x_i \geq 0 \qquad [7]$$

R Time!

https://github.com/MatthewALanham/2025_informs/

```r
# lets review our LP model
lps.model
# solve the model
solve(lps.model)
get.objective(lps.model) # optimal obj. value (i.e. our maximum profit)
get.variables(lps.model) # optimal solution of d.v.'s (our decisions to make)
```

```
> get.objective(lps.model) # optimal obj. value (i.e. our maximum profit)
[1] 30.20487
> get.variables(lps.model) # optimal solution of d.v.'s (our decisions to make)
 [1] 1.0000000 9.1666667 6.6666667 0.8333333 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
[10] 0.1666667
```

```
> (results <- rbind(results,results2))
                                Model        x1       x2    Exp[y] Exp[Cost]
1                        Max[Quality] 22.500000 0.000000 63.09058       225
2      Max[Quality] + Min-Max Constraints 10.000000 6.250000 32.26023       225
3 Max[Quality] + Convex Hull Constraints  9.166667 6.666667 30.20487       225
```

https://github.com/MatthewALanham/2025_informs/

Stage 2: Other optimization model formulations

Another possible formulation that could reasonably occur in practice would be to incorporate the predictive model as a constraint into the optimization model, rather than directly formulating it into the optimization model's objective function.

Model formulation 4: Minimize cost

$$\min \ \$10x_1 + \$20x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq 225 \quad [3]$$

$$x_i \geq 0 \quad [4]$$

Model formulation 5:

Minimize cost with min-max constraint

$$\min \ \$10x_1 + \$20x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq 225 \quad [3]$$

$$\min(x_1) \leq x_1 \leq \max(x_1) \quad [4]$$

$$\min(x_2) \leq x_2 \leq \max(x_2) \quad [5]$$

$$x_i \geq 0 \quad [6]$$

Model formulation 6:

Minimize cost with convex hull constraint

$$\min \ \$10x_1 + \$20x_2 \quad [1]$$

subject to:

$$2.674 + 2.685x_1 + 0.437x_2 \geq 29 \quad [2]$$

$$\$10x_1 + \$20x_2 \leq \$225 \quad [3]$$

$$\sum_{i=1}^{J} w_j v_{j1} = x_1 \quad [4]$$

$$\sum_{i=1}^{J} w_j v_{j2} = x_2 \quad [5]$$

$$\sum_{i=1}^{J} w_j = 1 \quad [6]$$

$$x_i \geq 0 \quad [7]$$

Comparison of solutions

Solving the maximizing stability model formulations and comparing them to the minimizing cost model formulations.

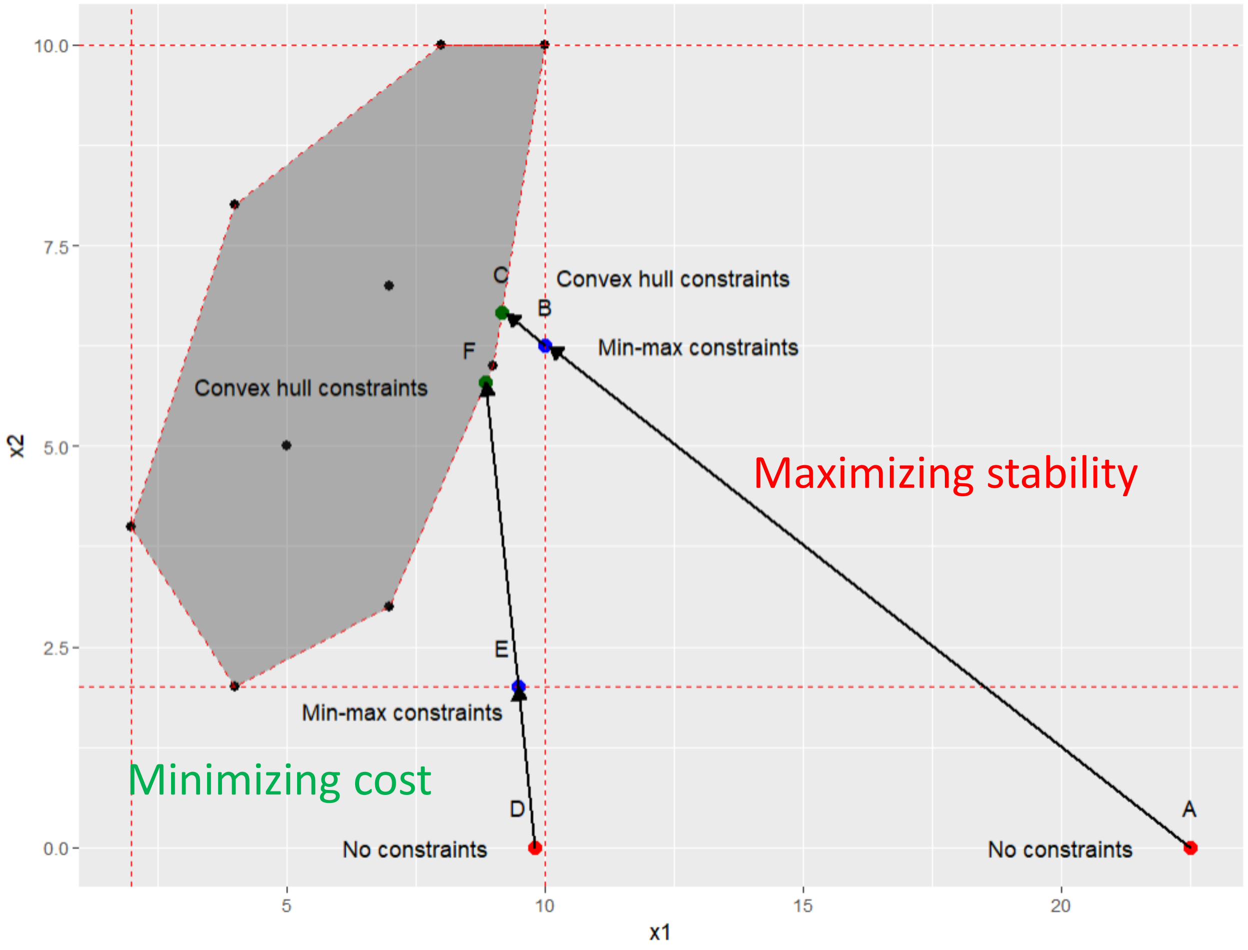
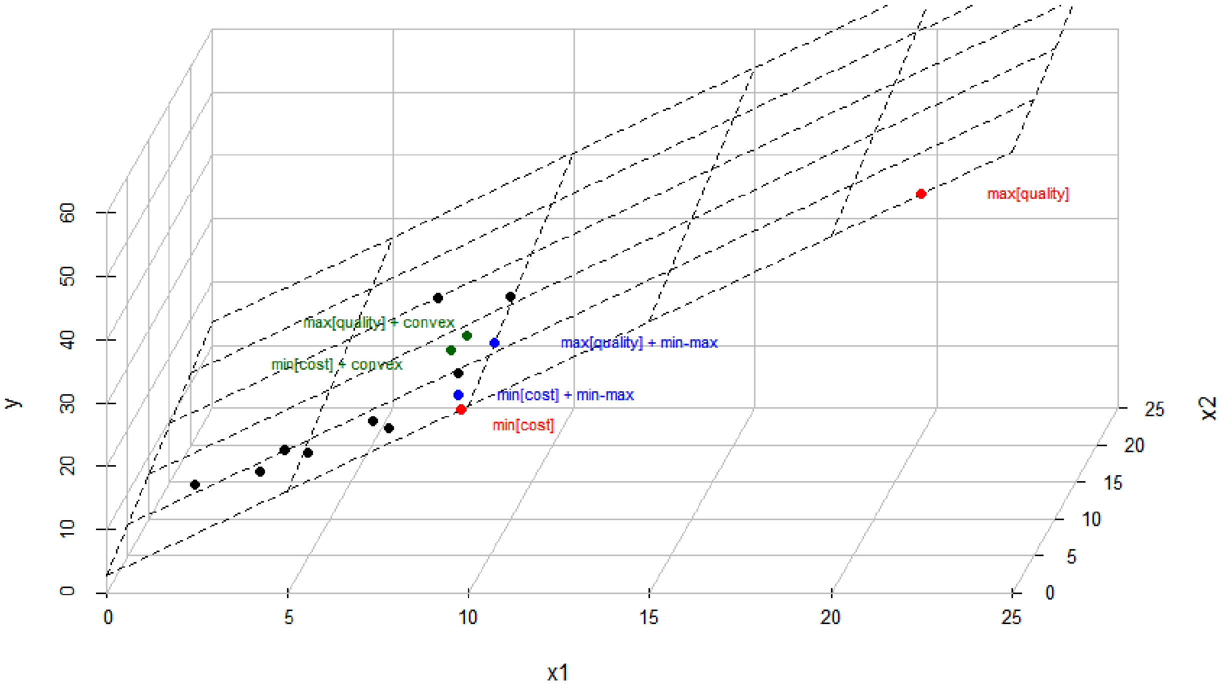


Figure 12: Visualization of six solved model formulations

How did the constraints perform?

- Both formulations using the estimated predictive model in either objective function or constraint had an extrapolated prediction or recommendation.
- Both formulations using convex hull constraints led to “data supported” decisions recommendations
- Now, you must consider the risk of the expected quality (y). Remember it could be higher or lower (it’s an expected value).



	Point ID	x1	x2	Exp[y]	Exp[Cost]
Max[Quality]	A	22.5	0	63.09	225
Max[Quality] + Convex Hull Constraints	B	9.17	6.67	30.2	225
Max[Quality] + Min-Max Constraints	C	10	6.25	32.26	225
Min[Cost]	D	9.8	0	29	98.04
Min[Cost] + Min-Max Constraints	E	9.48	2	29	134.78
Min[Cost] + Convex Hull Constraints	F	8.86	5.79	29	204.42

Table 4: Summary of model formulation experiments

First, lets say the problem or data is just a bit bigger….

This problem we have been toying with is just a small sample from a carpet manufacturing process, where the technician is trying to find the best settings of three confidential chemical inputs we refer to as $X_1, X_2,$ and X_3. We have to achieve at least 80% stability…

Fig. Synthetic fibers coating process

- We know carpet stability is considered the primary KPI.
- Stability is a measure of the fibers after being stored in a warehouse for six months.
- Retailers require that stability be greater than or equal to 80% for all carpet types
- The dataset contains 150 different chemical experiments from an established process.

First stage – develop a predictive model

Input features

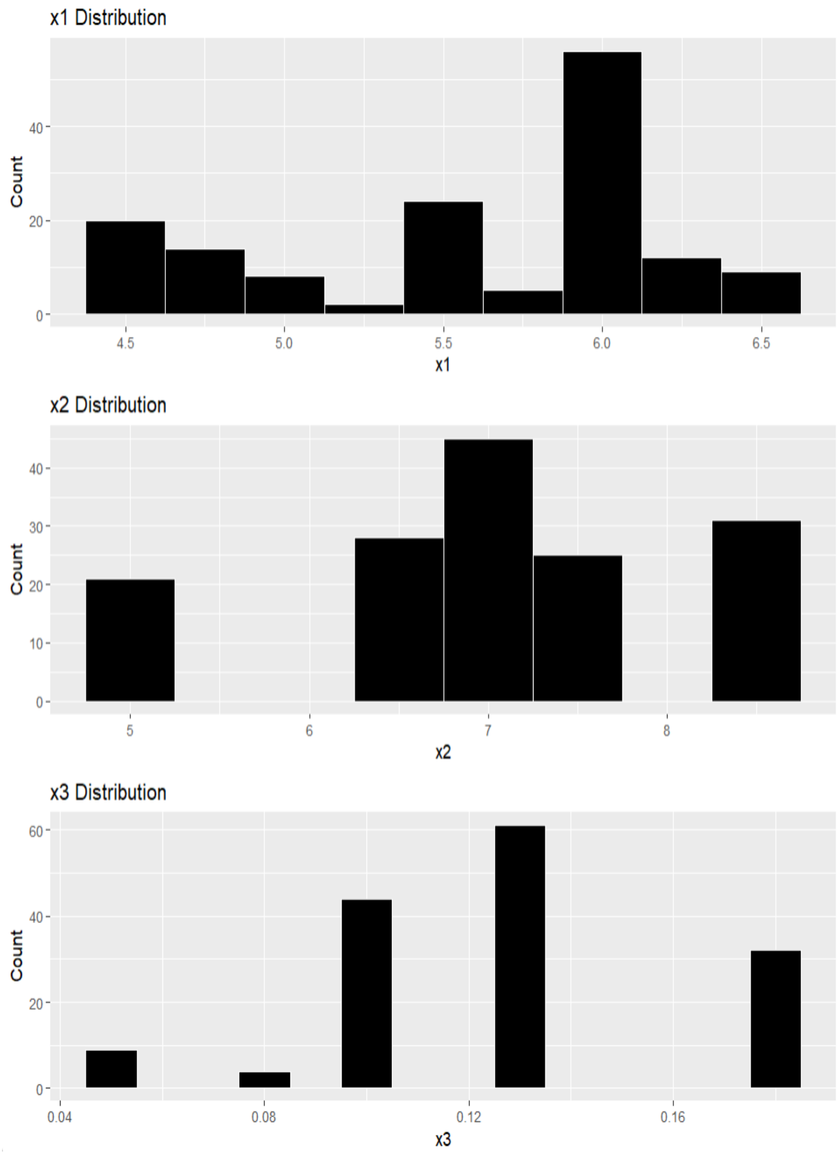
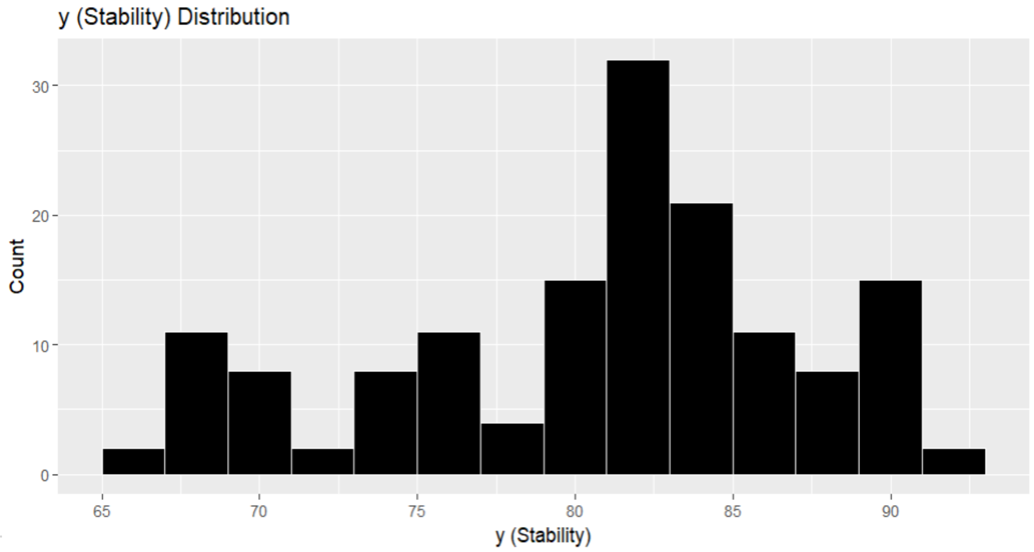
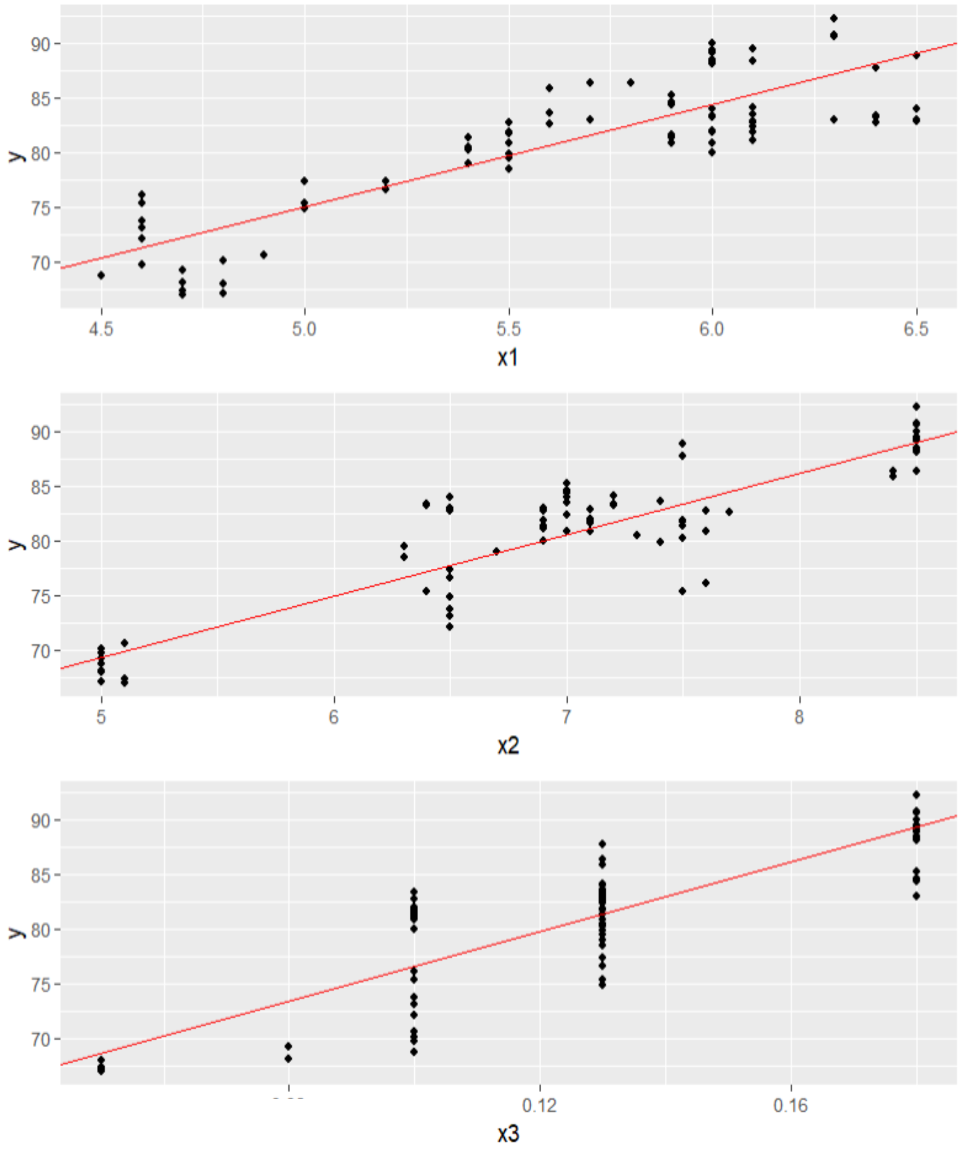


Fig. Feature distributions

Target variable



Descriptive analysis of data



	x1	x2	x3	y (Stability)	VIF
x1	1	0.59	0.63	0.87	1.758
x2	0.59	1	0.72	0.89	2.223
x3	0.63	0.72	1	0.84	2.404
y (Stability)	0.87	0.89	0.84	1	

Table 9. Pearson correlation matrix and Variance Inflation Factors (VIFs)

While the correlations among the input features are moderate, calculating variance inflation factors suggest the correlations should not negatively affect the estimation of the linear model.

A general rule of thumb is if VIFs are greater than 10 this should be a cause for concern, while more conservative estimates suggest VIFs greater than 2.5 (Dodge 2008). As shown in Table 9 the largest VIF among our input features is 2.404.

Dodge, Y. (2008). The Concise Encyclopedia of Statistics, Springer New York.

Linear regressions

The model chosen uses just the main effects. Has the highest adjusted R^2 of 0.988.

	Bx0	px0	Bx1	px1	Bx2	px2	Bx3	px3	Bx1x2	px1x2	Bx1x3	px1x3	Bx2x3	px2x3	Bx1x2x3	px1x2x3	AdjR2
1	28.102	0	9.399	0.00													0.751
2	41.204	0			5.62	0.00											0.782
3	60.689	0					159.299	0.00									0.702
4	22.912	0	5.753	0.00	3.641	0.00											0.968
5	35.007	0	6.093	0.00			92.293	0.00									0.895
6	44.695	0			3.703	0.00	79.423	0.00									0.866
7	27.139	0	5.005	0.00	2.883	0.00	42.066	0.00									0.988
8	43.534	0	1.871	0.09	0.568	0.52			7.42	1.69							0.97
9	22.056	0	8.402	0.00			213.358	0.00			-21.185	0.03					0.897
10	39.877	0			4.403	0.00	123.179	0.00					-6.155	0.18			0.866
11	27.118	0	5.007	0.00	2.885	0.00	42.175	0.00							-0.003	0.99	0.988

Table 8: Linear regression estimated model parameters, coefficient p-values, and adjusted R^2

$$\boxed{\hat{y} = 27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3}$$

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 27.13912    0.60436   44.91   <2e-16 ***
x1           5.00484    0.12636   39.61   <2e-16 ***
x2           2.88324    0.08325   34.63   <2e-16 ***
x3          42.06552    2.58881   16.25   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.717 on 146 degrees of freedom
Multiple R-squared:  0.9887,    Adjusted R-squared:  0.9885
F-statistic:  4261 on 3 and 146 DF,  p-value: < 2.2e-16
```

https://github.com/MatthewALanham/2025_informs/

Linear regression model diagnostics

There were no influential points identified in this linear model as shown in the Cook's Distance chart (Kutner, et. al. 2005) and Hadi's influence measure (Hadi, 1992)

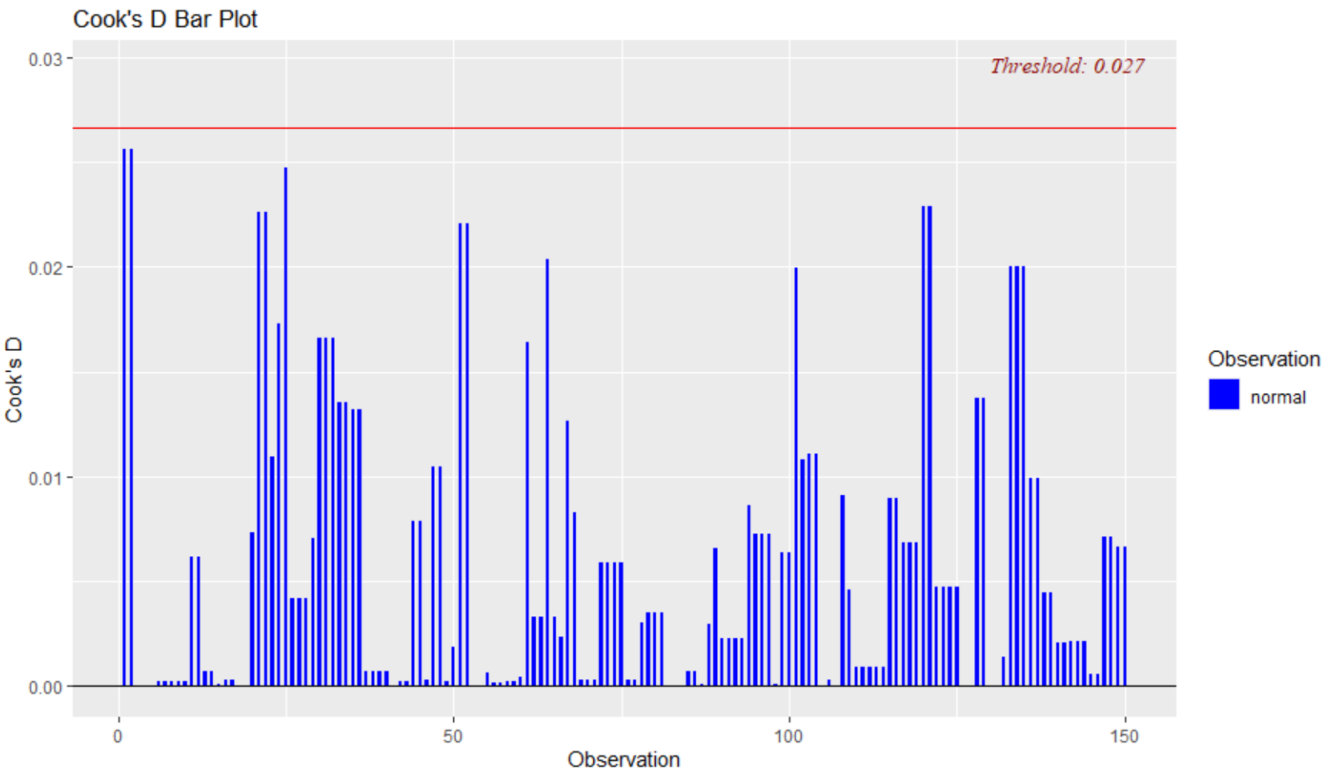


Figure 21. Cook's Distance

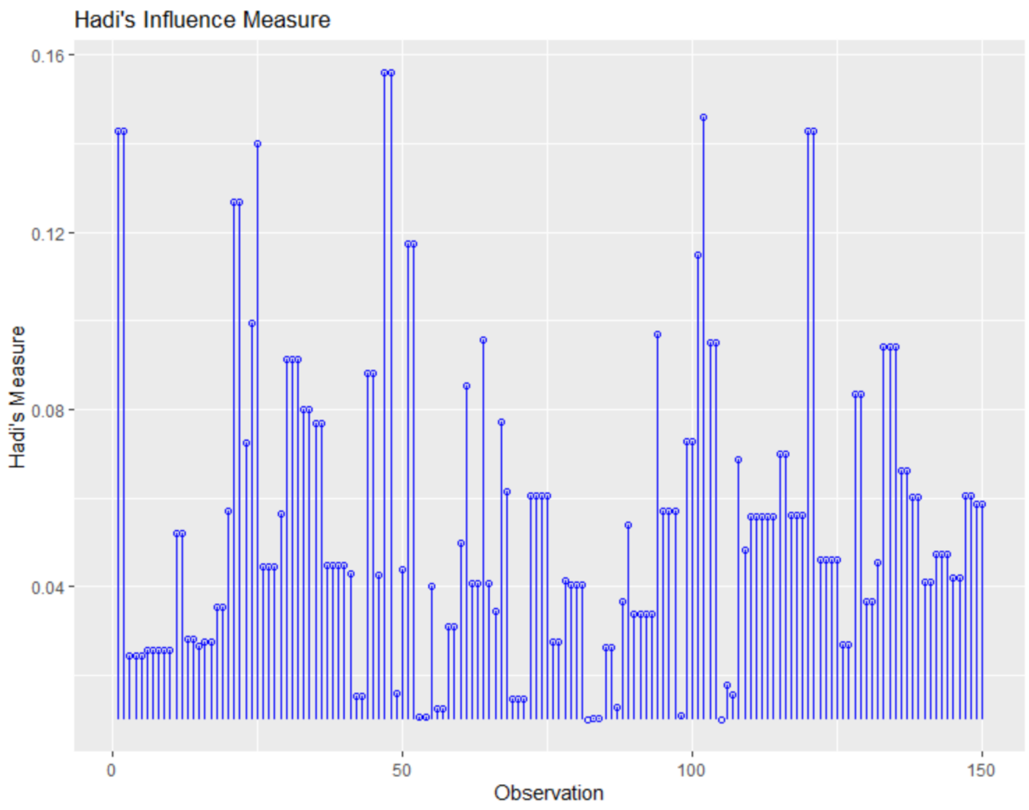


Figure 22. Hadi's influence measure

Kutner, M. H. (2005). Applied linear statistical models. Boston, McGraw-Hill Irwin.

Hadi, A. S. (1992). "A new measure of overall potential influence in linear regression." Computational Statistics & Data Analysis **14**(1): 1-27.

Linear regression model diagnostics

Residuals were normally distributed and homoskedastic (constant variance).

H_0: $variance\ is\ homogenous$
H_a: $variance\ is\ not\ homogenous$

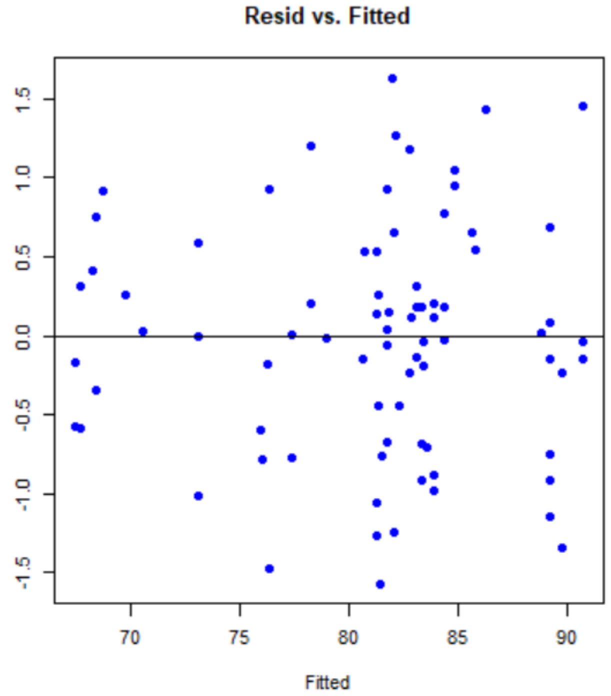
	Test Statistic	p-value
Breusch Pagan Test	1.0047	0.316
Score Test	1.2913	0.255
F-test	1.2852	0.258

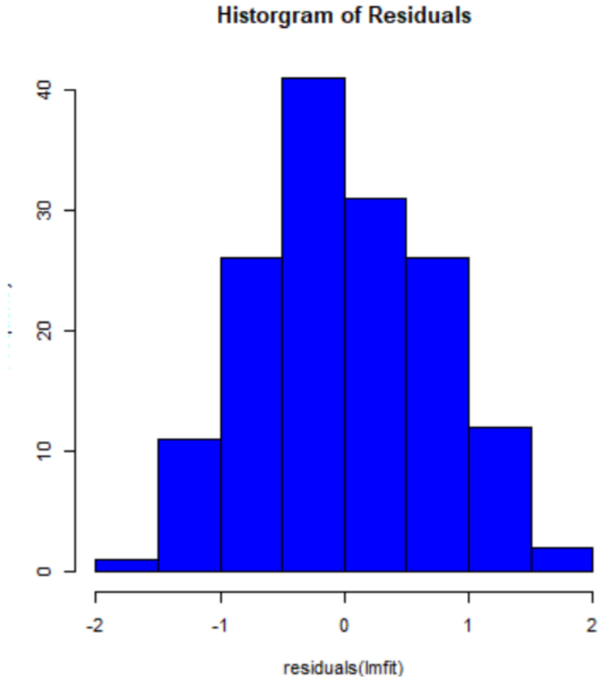
Table 10. Heteroskedasticity tests

H_0: $errors\ are\ normally\ distributed$
H_a: $errors\ are\ not\ normally\ distributed$

	Test Statistic	p-value
Shapiro-Wilk Test	0.98771	0.208
Lilliefors (Kolmogorov-Smirnov) Test	0.06870	0.080
Anderson-Darling Test	0.53022	0.173

Table 11. Normality tests





https://github.com/MatthewALanham/2025_informs/

Stage 2: optimization model formulation

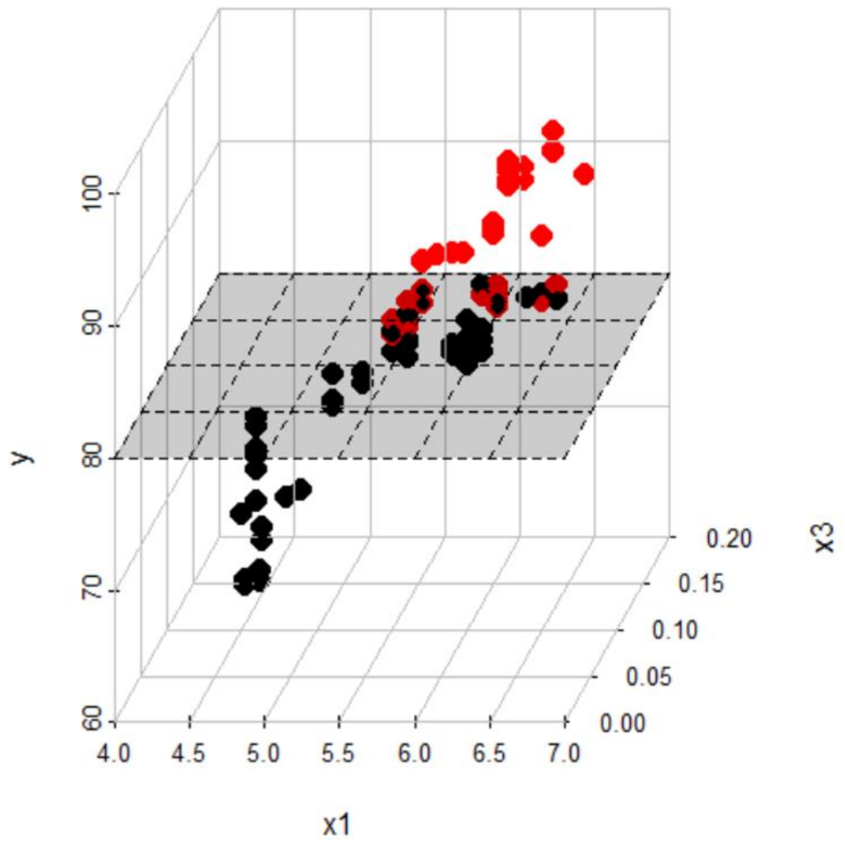


Figure 24. Scenarios that meet stability requirements (red) and do not meet stability requirements (black)

Formulating and solving this problem using the known business constraints of having a stability greater than 80 and staying within budget of $1000 leads to a decision of:

$x_1^* = 390.6$

$x_2^* = 0$

$x_2^* = 0$

$E[y] = 1982.15$

Model formulation 8: Maximize stability

$$\max\ y = 27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \quad [1]$$

s.t.

$$27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \geq 80 \quad [2]$$

$$\$2.56x_1 + \$2.50x_2 + \$21.52x_3 \leq \$1000 \quad [3]$$

$$x_i \geq 0 \quad [4]$$

Note: The costs associated with each input/decision variable are: $x_1 = \$2.56$, $x_2 = \$2.50$, $x_1 = \$21.52$

Stage 2: model formulation with min-max box constraints

Formulating and solving this problem using the known business constraints of having a specificity greater than 80 and staying within budget of $1000 leads to a decision of:

$x_1^* = 6.5$

$x_2^* = 8.5$

$x_2^* = 0.18$

$E[y] = 91.75$

Model formulation 9: Maximize stability w/ box constraints

max $y = 27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3$ [1]

s.t.

$27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \geq 80$ [2]

$\$2.56x_1 + \$2.50x_2 + \$21.52x_3 \leq \$1000$ [3]

$\min(x_1) \leq x_1 \leq \max(x_1)$ [4]

$\min(x_2) \leq x_2 \leq \max(x_2)$ [5]

$\min(x_3) \leq x_3 \leq \max(x_3)$ [6]

$x_i \geq 0$ [7]

https://github.com/MatthewALanham/2025_informs/

Stage 2: model formulation with convex hull constraints

Formulating and solving this problem using the known business constraints of having a specificity greater than 80 and staying within budget of $1000 ($37.93) leads to a decision of:

$x_1^* = 6.4$

$x_2^* = 7.5$

$x_2^* = 0.13$
$E[y] = 86.26$

Model formulation 3: Maximize stability w/ convex hull constraints

$$\max\ y = 27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \quad [1]$$

s.t.

$$27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \geq 80 \quad [2]$$

$$\$2.56x_1 + \$2.50x_2 + \$21.52x_3 \leq \$1000 \quad [3]$$

$$-x_1 + 6.4w_1 + 5.7w_2 + 5.7w_3 + 6.1w_4 + 4.8w_5 = 0 \quad [4]$$

$$-x_2 + 7.5w_1 + 8.5w_2 + 6.9w_3 + 7.1w_4 + 5.0w_5 = 0 \quad [5]$$

$$-x_3 + 0.13w_1 + 0.13w_2 + 0.18w_3 + 0.10w_4 + 0.10w_5 = 0 \quad [6]$$

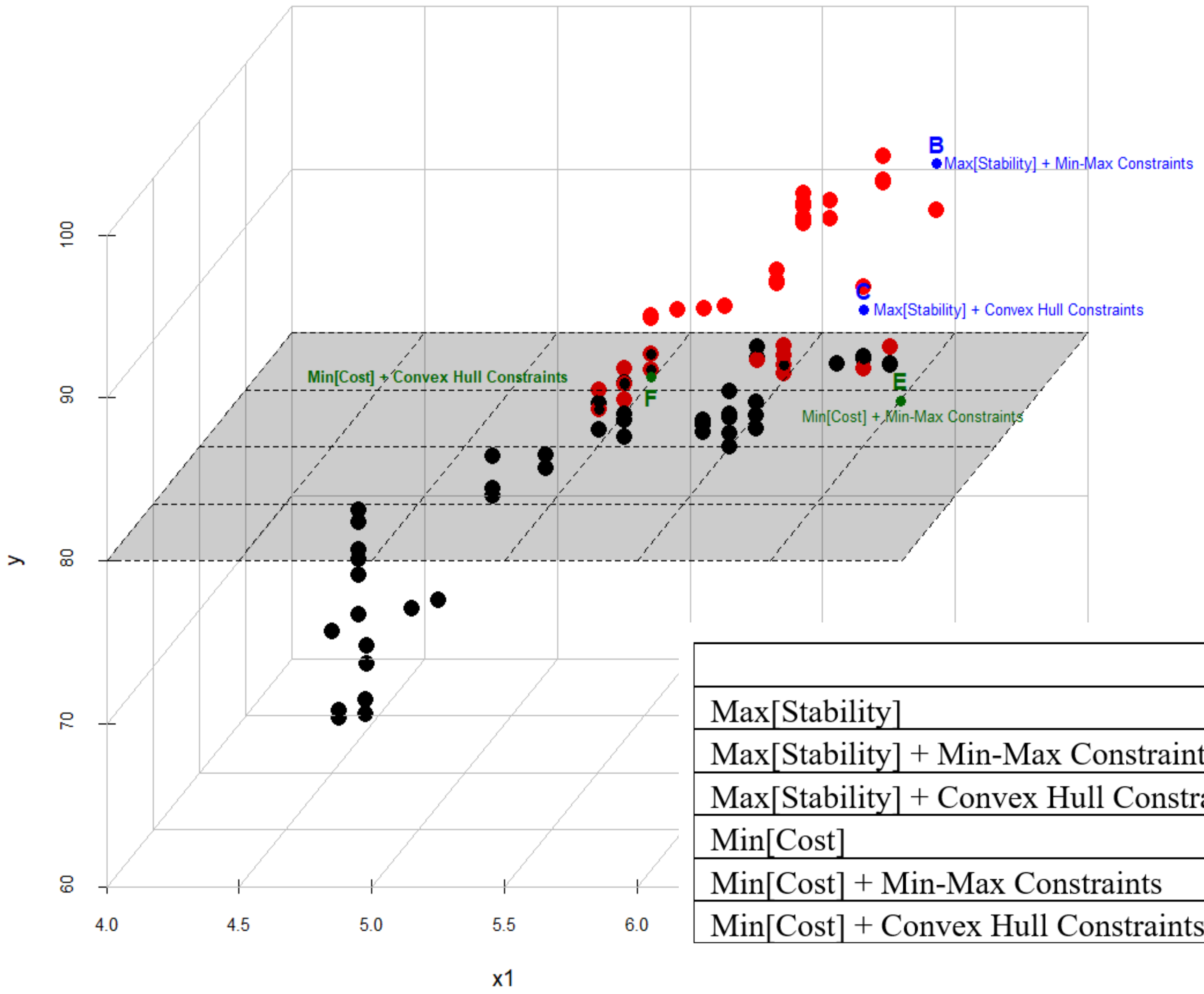
$$w_1 + w_2 + w_3 + w_4 + w_5 = 1 \quad [7]$$

$$w_j, x_i \geq 0 \quad [8]$$

```
> vertices
     x1  x2   x3
25   6.4 7.5 0.13
29   5.7 8.5 0.13
60   5.7 6.9 0.18
78   6.1 7.1 0.10
132  4.8 5.0 0.10
```

https://github.com/MatthewALanham/2025_informs/

Some experimental results



	x1	x2	x3
Minimum	4.50	5.00	0.05
Average	5.60	7.03	0.13
Median	5.90	7.10	0.13
Maximum	6.50	8.50	0.18

Table : Feature summary statistics

- Outside historical values + severe extrapolation
- Solution resides on box
- Outside historical values
- Hitting stability requirement in all cases

	Point ID	x1	x2	x3	Exp[y]	Exp[Cost]
Max[Stability]	not shown	390.62	0	0	1982.16	1000
Max[Stability] + Min-Max Constraints	B	6.5	8.5	0.18	91.75	41.76
Max[Stability] + Convex Hull Constraints	C	6.4	7.5	0.13	86.26	37.93
Min[Cost]	not shown	10.56	0	0	80	27.04
Min[Cost] + Min-Max Constraints	E	6.5	5	0.14	80	32.17
Min[Cost] + Convex Hull Constraints	F	5.49	6.45	0.16	80	33.66

Table 13: Experimental results

Identify and Hedge Where There is Uncertainty

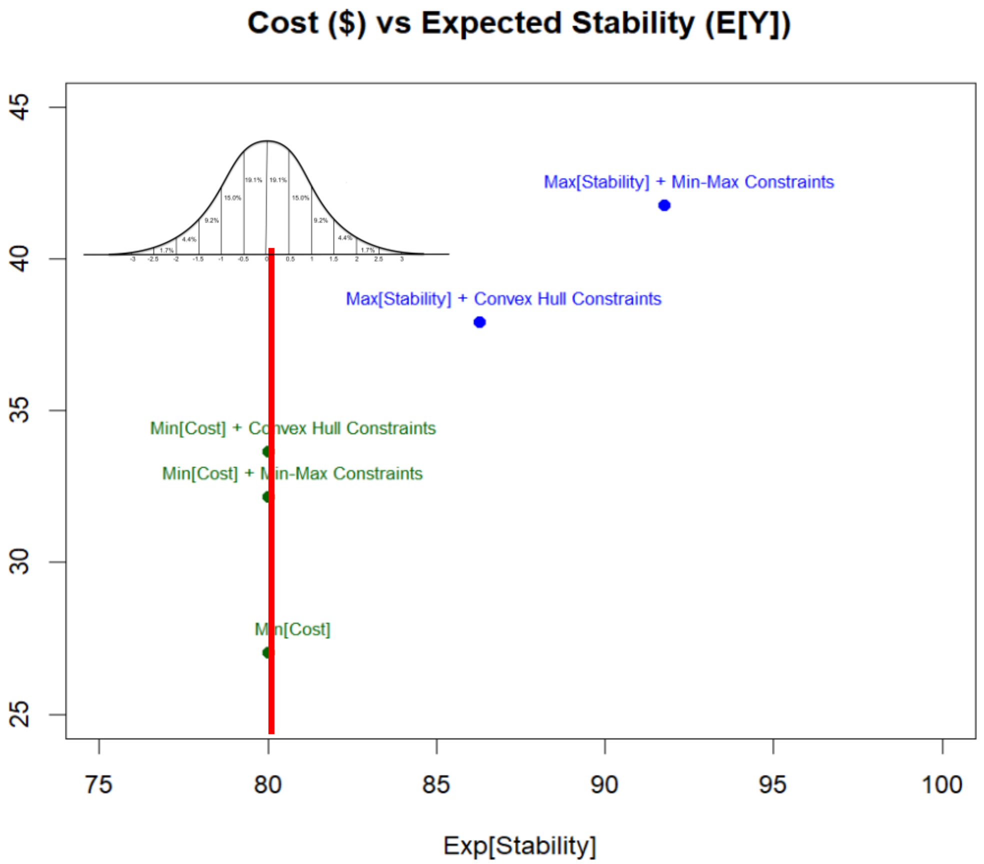
(where your predictive model(s) are)

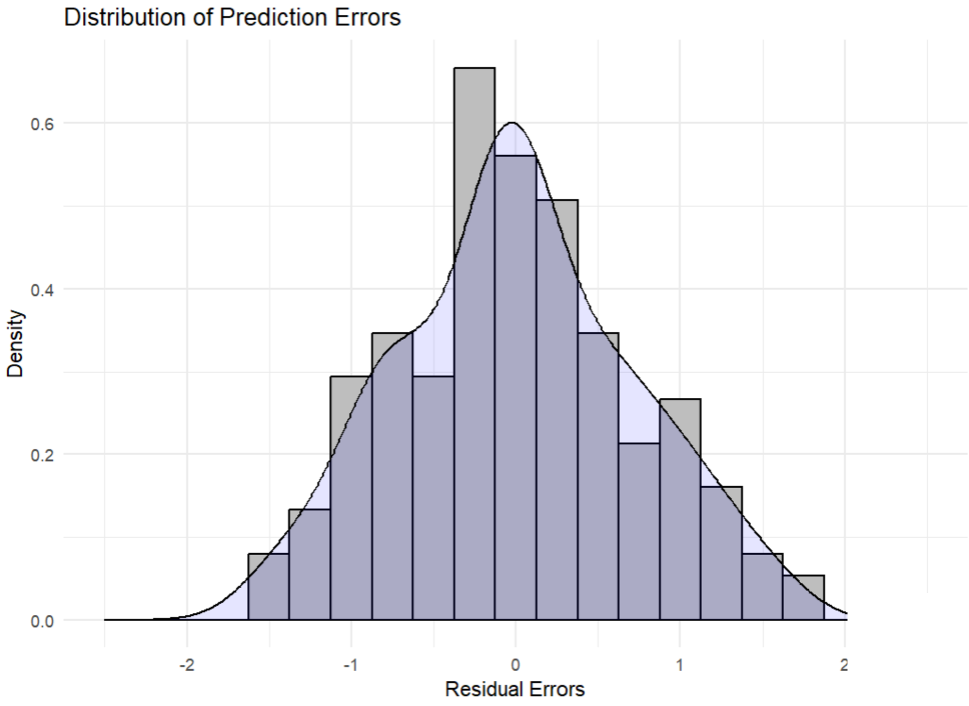
Professor Matthew Lanham
(Former) Academic Director, MS BAIM Program
Associate Director of Student Engagements,
Krenicki Center for Business Analytics & Machine Learning
MatthewALanham.com

Addressing uncertainty in your prediction

If we use these input settings for x_1, x_2, x_3, we might manufacturer a carpet where some portion of carpet actually has a specificity below 80!

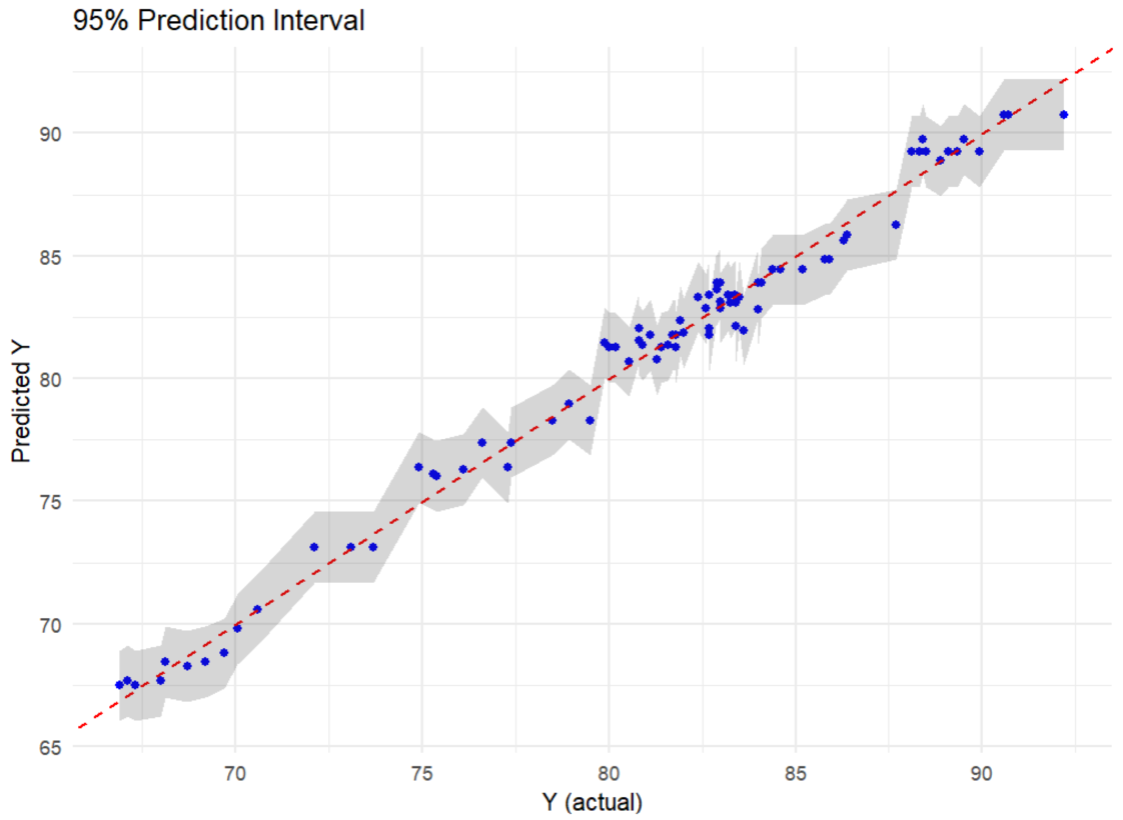
Comparing the predictions to historical values, 21.3% of the time when predicted stability is 80 or greater, the actual stability was less than 80.

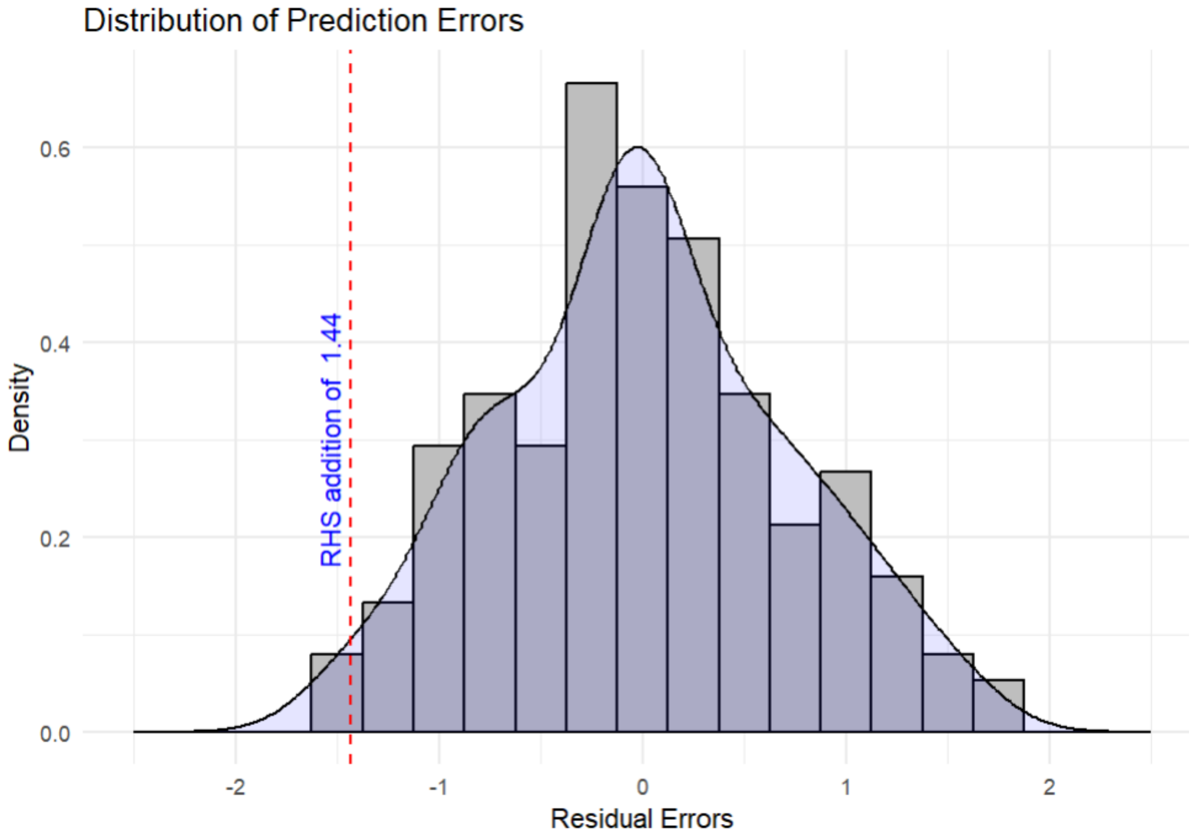




Possibly use a prediction interval adjustment

Estimating the prediction interval from the historical data, we would want to add 1.44 to the RHS of the stability constraint of 80 to any of our model formulations.





Add or subtract the PI bound based on the constraint type

PI_{ub}: *the prediction interval upper bound increment*

$\widehat{PI}_{ub} = 1.44$

Model formulation 11: Maximize stability w/ convex hull constraints + PI adjustments

$$\max\ y = 27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \quad [1]$$

s.t.

$$27.139 + 5.005x_1 + 2.883x_2 + 42.066x_3 \geq 80 + PI_{ub} \quad [2]$$

$$\$2.56x_1 + \$2.50x_2 + \$21.52x_3 \leq \$1000 \quad [3]$$

$$\sum_{i=1}^{J} w_j v_{j1} = x_1 \quad [4]$$

$$\sum_{i=1}^{J} w_j v_{j2} = x_2 \quad [5]$$

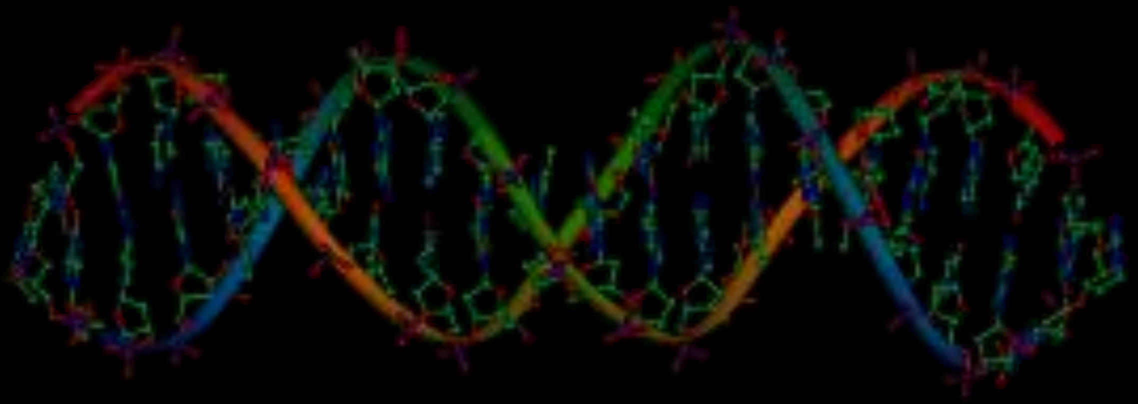
$$\sum_{i=1}^{J} w_j v_{j3} = x_3 \quad [6]$$

$$\sum_{i=1}^{J} w_j = 1 \quad [7]$$

$$w_j, x_i \geq 0 \quad [8]$$

Note: In our case, we are willing to have at most 5% of our carpet have a predicted stability less than 80. If that is too much risk for your situation, estimate the prediction interval increment to a higher number (e.g., 99.9 PI).

Going from parametric to non-parametric models



Professor Matthew Lanham
(Former) Academic Director, MS BAIM Program
Associate Director of Student Engagements,
Krenicki Center for Business Analytics & Machine Learning
MatthewALanham.com

When you have a machine learning model consider a GA

When we do not have nice parametric models anymore (e.g., linear regression, logistic regression), you are not going to be able to simply pull those β coefficients from stage 1 to put them in the stage 2 optimization. You might have some machine learning model – consider using a GA!

```
################################################################################
# Optimization model using genetic algorithm
################################################################################
# Recall the linear regression the model
f <- lm(y ~ x1 + x2 + x3, data=d)
summary(f)

# Define the coefficients from the linear regression model
(coefficients <- coef(f))

# Objective function to maximize
objective_function <- function(params) {
  # params are x1, x2, x3
  intercept <- coefficients[1]
  x1 <- params[1]
  x2 <- params[2]
  x3 <- params[3]
  # Calculate the expected value of y
  expected_y <- (intercept + coefficients[2]*x1 + coefficients[3]*x2
                 + coefficients[4]*x3)
  # Return the expected value of y
  return(expected_y)
}
```

We could just create our own objective function as a function.

Set predictive model as our fitness (objective) function

Whatever GA library you use, it will ask you to set box constraints using the lower and upper bounds. This is our solution search space. We have learned we can create custom search spaces (e.g., convex hull).

Using the GA library

Objective function

Min-max constraints

```
# Define the bounds for x1, x2, and x3
lower_bounds <- c(min(d$x1), min(d$x2), min(d$x3))
upper_bounds <- c(max(d$x1), max(d$x2), max(d$x3))

# Run the genetic algorithm
library(GA)
# Set seed for reproducibility
set.seed(123)
ga_result <- ga(
  type = "real-valued",
  fitness = objective_function,
  lower = lower_bounds,
  upper = upper_bounds,
  popSize = 50,      # Population size
  maxiter = 100,     # Maximum number of iterations
  run = 50           # Number of generations without improvement
)

# Summary of GA results
summary(ga_result)

# Best solution found
best_solution <- ga_result@solution

# Calculate the maximum expected value of y
max_expected_y <- objective_function(best_solution)
```

GA found a similar solution but not exactly the same

The package will print out each iteration as it runs. You'll notice in this case the max stability with box (min-max) constraints model is similar to what we obtained previously.

```
GA | iter = 88 | Mean = 90.26375 | Best = 90.93253
GA | iter = 89 | Mean = 90.01561 | Best = 90.93253
GA | iter = 90 | Mean = 89.92768 | Best = 90.93253
GA | iter = 91 | Mean = 89.95010 | Best = 90.93253
GA | iter = 92 | Mean = 89.79621 | Best = 90.93253
GA | iter = 93 | Mean = 89.94559 | Best = 90.93253
GA | iter = 94 | Mean = 89.95083 | Best = 90.93253
GA | iter = 95 | Mean = 90.08467 | Best = 90.93253
GA | iter = 96 | Mean = 90.07542 | Best = 90.93253
GA | iter = 97 | Mean = 90.24760 | Best = 90.93253
GA | iter = 98 | Mean = 89.84782 | Best = 90.93253
GA | iter = 99 | Mean = 89.90236 | Best = 90.93253
GA | iter = 100 | Mean = 90.17818 | Best = 90.93253
>
```

```
> # Print results
> print(paste("Best solution (x1, x2, x3):", paste(best_solution, collapse = ", ")))
[1] "Best solution (x1, x2, x3): 6.48085990877272, 8.3833814779108 4, 0.17083942249299 5"
> print(paste("Maximum expected value of y:", max_expected_y))
[1] "Maximum expected value of y: 90.9325291928572"
```

Max[Stability] + Min-Max Constraints	B	6.5	8.5	0.18	91.75

Lets predict with ML in S1, then optimize using GA in S2

Used the caret library to train the models using a 3-fold CV, but it does not really matter what you use here.

1. Random Forest
2. Support Vector Regression
3. XG-Boost
4. Neural Net

```
################################################################################
# Integrating a machine learning models into an optimization model using a
# genetic algorithm
################################################################################
# Set seed for reproducibility
set.seed(123)
library(caret)
# defined a 3-fold cross-validation design
ctrl <- trainControl(method="cv",      # cross-validation set approach to use
                     number=3,          # k number of times to do k-fold
                     classProbs = F,    # if you want probabilities
                     #summaryFunction = twoClassSummary, # for classification
                     summaryFunction = defaultSummary,  # for regression
                     allowParallel=T)
################################################################################
# Train different models
################################################################################
# train various machine learning models
# https://topepo.github.io/caret/available-models.html
rf <- train(y~x1+x2+x3, data=d, method="rf", trControl=ctrl, metric="Rsquared")
svr <- train(y~x1+x2+x3, data=d, method="svmPoly", trControl=ctrl, metric="Rsquared")
xgb <- train(y~x1+x2+x3, data=d, method="xgbLinear", trControl=ctrl, metric="Rsquared")
ann <- train(y~x1+x2+x3, data=d, method="glmnet", trControl=ctrl, metric="Rsquared")
```

https://github.com/MatthewALanham/2025_informs/

Captured cross-validated performance

For this toy problem with nice linear relationships, that is also highly accurate, we did not need to use an ML model. Recall out linear regression model had an R^2 of 0.988.

```r
# final model
(ml_stats = data.frame(matrix(nrow=4, ncol=2, data=NA)))
(names(ml_stats) <- c("model","R2"))
top <- tolerance(rf$results, metric="Rsquared", tol=0.01, maximize=TRUE)
ml_stats[1,] <- c(model="rf", R2=rf$results[top,1:6][["Rsquared"]])
top <- tolerance(svr$results, metric="Rsquared", tol=0.01, maximize=TRUE)
ml_stats[2,] <- c(model="svr", R2=svr$results[top,1:6][["Rsquared"]])
top <- tolerance(xgb$results, metric="Rsquared", tol=0.01, maximize=TRUE)
ml_stats[3,] <- c(model="xgb", R2=xgb$results[top,1:6][["Rsquared"]])
top <- tolerance(ann$results, metric="Rsquared", tol=0.01, maximize=TRUE)
ml_stats[4,] <- c(model="ann", R2=ann$results[top,1:6][["Rsquared"]])
ml_stats$R2 <- as.numeric(ml_stats$R2)
ml_stats[,2] <- round(ml_stats[,2],4)
ml_stats
```

We observe similar predictive R^2 performance

1. Random Forest
2. Support Vector Regression
3. XG-Boost
4. Neural Net

```
> ml_stats
  model     R2
1    rf 0.9853
2   svr 0.9896
3   xgb 0.9860
4   ann 0.9878
```

Lets capture our stage 2 experimental results

Here I added in some columns to capture our experimental results for our 3 decision variables and they expected stability (a.k.a. “best_soln”)

```
ml_stats[,3:6] <- NA
names(ml_stats)[3:6] <- c("x1","x2","x3","best_soln")
ml_stats
ml_models <- ml_stats$model
```

```
> ml_stats
  model      R2 x1 x2 x3 best_soln
1    rf 0.9853 NA NA NA        NA
2   svr 0.9896 NA NA NA        NA
3   xgb 0.9860 NA NA NA        NA
4   ann 0.9878 NA NA NA        NA
> (ml_models <- ml_stats$model)
[1] "rf"  "svr" "xgb" "ann"
```

Lets run some experiments

```r
for (i in 1:4) {
# Objective function to maximize
objective_function <- function(params, model=get(ml_models[[i]])) {
  # params are x1, x2, x3
  x1 <- params[1]
  x2 <- params[2]
  x3 <- params[3]
  # Create a new data frame for prediction
  new_data <- data.frame(x1 = x1, x2 = x2, x3 = x3)
  # Predict y using the random forest model
  predicted_y <- predict(model, new_data)
  # Return the predicted value of y
  return(predicted_y)
}

# Define the bounds for x1, x2, and x3
lower_bounds <- c(min(d$x1), min(d$x2), min(d$x3))
upper_bounds <- c(max(d$x1), max(d$x2), max(d$x3))
```

Lets set up our stage 2 objective function. We'll just loop through and try our four different ML models

```r
# Run the genetic algorithm
ga_result <- ga(
  type = "real-valued",
  fitness = objective_function,
  lower = lower_bounds,
  upper = upper_bounds,
  popSize = 50,      # Population size
  maxiter = 100,     # Maximum number of iterations
  run = 50           # Number of generations without improvement
)

# Summary of GA results
summary(ga_result)

# Best solution found
best_solution <- ga_result@solution
best_solution <- best_solution[1,]
best_solution
ml_stats[i,3:5] <- best_solution

# Calculate the maximum expected value of y
max_expected_y <- objective_function(best_solution)
max_expected_y
ml_stats[i,6] <- max_expected_y

}
```

We set up the GA with min-max constraints

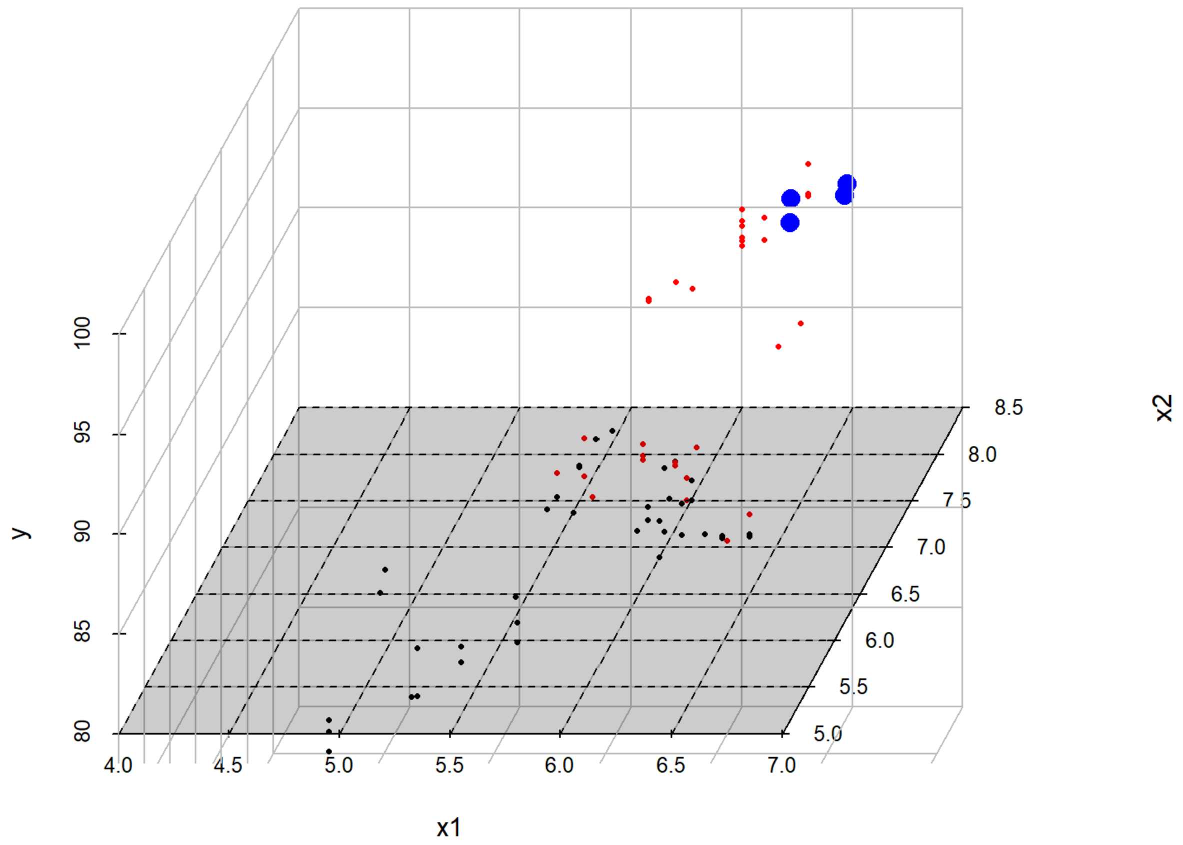
Capture solution and decision reccommendations

Iterations... final results!

```
GA | iter = 25 | Mean = 90.66990 | Best = 90.97535
GA | iter = 26 | Mean = 90.41697 | Best = 90.97535
GA | iter = 27 | Mean = 90.37301 | Best = 90.97535
GA | iter = 28 | Mean = 90.05512 | Best = 90.97535
GA | iter = 29 | Mean = 90.50967 | Best = 90.97535
GA | iter = 30 | Mean = 90.28027 | Best = 90.97535
GA | iter = 31 | Mean = 89.60918 | Best = 90.97535
```

```
> ml_stats
  model     R2       x1       x2        x3 best_soln
1    rf 0.9853 6.355228 8.450711 0.1644634  90.71081
2   svr 0.9896 6.418988 8.411746 0.1787441  91.20753
3   xgb 0.9860 6.241321 8.136208 0.1551439  90.97535
4   ann 0.9878 6.390735 8.363910 0.1738644  90.46661
```

Here we can see the various "best" solutions using a GA on top of each of the four ML models tried.



Take-Aways

https://github.com/MatthewALanham/2025_informs/

- In toy-sized problems you will likely be able to “see” where weird things are and account for them with custom constraints, but bigger problems beware – out of sight out of mind.
- Box constraints can help safeguard against extrapolating.
- Convex hull constraints might be better – based on how much you are willing to deviate from historical values.
- Do not forget to identify and hedge where the uncertainty lies (e.g., RHS adjustment).
- When you move from parametric-type models to non-parametric (ML), genetic algorithms (GA) can be an option.