Course Info

Prereqs: Three semesters of Calculus

Co-reqs: Linear Algebra (MAT 202 or MAT 204)


Grading:  
Homework: 25%  
Midterm 1: 25%  
Midterm 2: 25%  
Final Project: 25%

Homework:  
• Will be due every week at 9pm on Friday.  
• All homework must be submitted via Blackboard.  
• The lowest homework grade will be dropped.

Midterms: Midterms will be in-class on Thursday of the 6th and 11th weeks.

Lectures: Reading assignments will be posted in advance of each lecture. You should read the reading material before lecture.

Slides: The slides will be posted online. But, they are not a replacement for the lecture. They are just my notes to remind me what to say. You must go to lecture to hear what I have to say.

Engineering is the process of taking the discoveries from science... implementing them as practical devices, and then ...

making them better, ...

and better, ...

and better.

This is optimization.

In this class, we will take a more mathematical approach.

We will also use computational tools to solve numerically the practical problems we encounter.
Optimization via (Freshman) Calculus

Express an *objective function* to be *minimized* or *maximized* in terms of one independent variable.

Differentiate with respect to this variable.

Set derivative equal to zero.

Solve for the independent variable.

If in doubt as to whether it’s a max, min, or saddle point, take second derivative and look at its sign.

If the independent variable is restricted to lying in an interval of the real line, check the endpoints—the optimal solution could be there.
River Crossing
An Example: River Crossing

Suppose you are on one side of a river (at coordinates \((a_1, b_1)\) in the figure) and there is a treasure on the shore at the other side (at coordinates \((a_2, b_2)\)). Worried that someone else might get the treasure before you, you'd like to get there as fast as possible—in *minimum* time.

Assuming that your running speed is \(v_1\) and and your swimming speed is \(v_2\) and that you choose to reach the river at \((x, 0)\), the time is given by:

\[
T(x) = \frac{\sqrt{(a_1 - x)^2 + b_1^2}}{v_1} + \frac{\sqrt{(x - a_2)^2 + b_2^2}}{v_2}
\]

Derivative is:

\[
\frac{dT}{dx} = -\frac{1}{v_1} \frac{a_1 - x}{\sqrt{(a_1 - x)^2 + b_1^2}} + \frac{1}{v_2} \frac{x - a_2}{\sqrt{(x - a_2)^2 + b_2^2}} = 0.
\]

Rearranging, we get *Snell's Law*: \(n_1 \sin \theta_1 = n_2 \sin \theta_2\), where \(n_1 = 1/v_1\) and \(n_2 = 1/v_2\).
Solving it on the Computer (using AMPL)

# Getting to the treasure fast!
param info symbolic, := "File name: river_crossing.txt; Author: R.J. Vanderbei";
display info;

param a1; param b1;
param a2; param b2;
param v1; param v2;

var x;

minimize time: sqrt((a1-x)^2 + b1^2)/v1 + sqrt((x-a2)^2 + b2^2)/v2;

data;

param a1 := 60;
param b1 := 40;
param a2 := -40;
param b2 := -50;
param v1 := 10;
param v2 := 1.5;

solve;
display x;
Comments: The hashtag symbol (#) starts a “comment”.

Statements: Every “statement” ends with a semicolon (;).

Model Section: The first part of the code defines the problem without necessarily providing any specific data values.

Data: Data/parameters are introduced with the `param` command.

Variables: Variables are introduced with the `var` command.

Objective: The objective is to maximize or minimize a function. The function must be given a name followed by a colon (:) followed by a formula that defines the function.

Constraints: Constraints are introduced with the “subject to” command. Each constraint must be given a name followed by a colon (:) followed by the equality or inequality that defines the constraint.

Data Section: The data section starts with the `data` command. In this section parameter definitions are repeated and values are given.

Solve: The `solve` command invokes the solver to solve the problem.

Display: The `display` command is used to see the results.
• The language is called **AMPL**, which stands for *A Mathematical Programming Language*.

  Note: “Modern” optimization dates back to the 1940’s where it was a useful/important tool helping the military prepare their program of activities. Hence, it was called *Mathematical Programming* and if the problem was linear it was called *Linear Programming*. This terminology predates the field of *Computer Programming*. The modern trend (finally!) is to replace the word “programming” with “optimization”.

• The book describing the language is called “AMPL” by Fourer, Gay, and Kernighan. It is available for free at [http://www.ampl.com/BOOK/download.html](http://www.ampl.com/BOOK/download.html).


  There are links to these AMPL websites on the course webpage:


• There are also online tutorials:
  
  – Google “AMPL tutorial” for examples.
There are three ways to access AMPL:

**Online:** The *Network Enabled Optimization Server (NEOS).*


**Download Course Version:** Download from course-specific link to your own computer. Expires at the end of the semester. Unlimited number of variables/constraints. *Preferred method.*

Details about these three methods are available here:

# Getting to the treasure fast!

param info symbolic := "File name: river_crossing.txt; Author: R.J. Vanderbei"

display info;

param a1; param b1;
param a2; param b2;
param v1; param v2;

var x;

minimize time: sqrt((a1-x)^2 + b1^2)/v1 + sqrt((x-a2)^2 + b2^2)/v2;

data;

param a1 := 60;
param b1 := 40;
param a2 := -40;
param b2 := -50;
param v1 := 10;
param v2 := 1.5;

solve;

display x;

reset;
To start using *jupyter* in your favorite browser, type this into a *Terminal* window...

Here’s the python notebook for the river crossing problem:

http://www.princeton.edu/~rvdb/307/python/RiverCrossing.ipynb
from amplpy import AMPL
ampl = AMPL()  # this creates an ampl object using your ampl installation
ampl.setOption('solver', 'loqo')  # this selects solver used for problem

# articulate optimization problem
ampl.eval('''
  # Getting to the treasure fast!
  param info symbolic, := "File name: RiverCrossing.ipynb; Author: R.J. Vanderbei";
  display info;
  param a1; param b1;
  param a2; param b2;
  param v1; param v2;
  var x;
  minimize time: sqrt((a1-x)^2 + b1^2)/v1 + sqrt((x-a2)^2 + b2^2)/v2;
  data;
  param a1 := 60;
  param b1 := 40;
  param a2 := -40;
  param b2 := -50;
  param v1 := 10;
  param v2 := 1.5;
  solve;
  display x;
''')

info = 'File name: RiverCrossing.ipynb; Author: R.J. Vanderbei'

LOQO 7.03: optimal solution (10 iterations, 10 evaluations)
primal objective 43.78213699
dual objective 43.78213699
x = -33.0434
NEOS is the *Network Enabled Optimization Server* supported by our federal government and located at the *University of Wisconsin*.

To submit an AMPL model to NEOS...

- **visit**: [http://www.neos-server.org/neos/](http://www.neos-server.org/neos/),
- **click**: on the *Submit a job to NEOS*,
- **scroll**: to the *Nonlinearly Constrained Optimization* list,
- **click**: on *LOQO [AMPL input]*,
- **scroll**: to *Commands File:*,
- **click**: on *Choose File*,
- **select**: a file from your computer that contains an AMPL model,
- **scroll**: to *e-mail address:*,
- **type**: your email address, and
- **click**: *Submit to NEOS*.

Piece of cake!
First Problem of First Assignment

Suppose the treasure is not exactly at the shore but rather is a certain distance away from the river. As shown, we are assuming the north shore of the river runs along the $x$-axis of our coordinate system. Assume that the river is $w = 30$ meters wide. For this problem, we need to figure out two things: (i) where you should enter the river and (ii) where you should exit it.

Write an AMPL model for this problem. Solve the problem using

$$
(a_1, b_1) = (60, 40) \\
(a_2, b_2) = (-50, -50) \\
v_1 = 10 \\
v_2 = 1.5 \\
v_3 = 7
$$

Report the $x$-coordinate of the location at which you should enter the river and the $x$-coordinate of the location at which you should exit the river.
Freshman Calculus

- One variable
- Nonlinear objective function
- Sometimes variable constrained to an interval

ORF 307

- Thousands of variables
- Linear objective function
- Linear (equality and inequality) constraints

There are multiple objectives for this course.

- Gain experience in formulating real-world problems as optimization problems.
- Learn how to distinguish good formulations from not-so-good ones.
- Learn how to solve real-world problems using AMPL software.
- Learn/understand the algorithms one uses to solve the problems.
Diet Problem
The McDonald’s Diet Problem

In words:

Minimize:

Calories

Subject to:

Total amounts of nutrients fall between certain minimum and maximum values.
# --- Declare the data sets and parameters ---------------

set NUTR;
set FOOD;

param f_min {FOOD} >= 0, default 0;
param f_max {j in FOOD} >= f_min[j], default 200;

param nutr_ideal {NUTR} >= 0;

param amt {NUTR,FOOD} >= 0;

# --- Declare the variables ------------------------------

var Buy {j in FOOD} integer >= f_min[j], <= f_max[j];

# --- State the objective --------------------------------

minimize Calories: sum {j in FOOD} amt["Cal",j] * Buy[j];

# --- State the constraints ------------------------------

subject to Dietary_bounds {i in NUTR}:
    0.8*nutr_ideal[i] <= sum {j in FOOD} amt[i,j] * Buy[j] <= 1.2*nutr_ideal[i];
The Data

param: NUTR: nutr_ideal :=
Cal 2500
CalFat 600
Fat 65
SatFat 20
Chol 300
Sodium 2400
Carbo 300
Protein 50
VitA 100
VitC 100
Calcium 100
Iron 100

;

set FOOD :=
"Bacon_Buffalo_Ranch_McChicken 5.6_oz_(159_g)"
"Big_Breakfast_(Large_Size_Biscuit) 10_oz_(283_g)"
"Big_Mac 7.6_oz_(215_g)"
"Chicken_McNuggets_(10_piece) 5.7_oz_(162_g)"
"Coca-Cola_Classic_(Medium) 21_fl_oz_cup"
"Diet_Coke_(Medium) 21_fl_oz_cup"
"Double_Quarter_Pounder_with_Cheese++ 10_oz_(283_g)"
"Egg_McMuffin 4.8_oz_(135_g)"
"Frappe_Caramel_(Medium) 16_fl_oz_cup"
"Hamburger 3.5_oz_(100_g)"
"Hash_Brown 2_oz_(56_g)"
"Large_French_Fries 5.4_oz_(154_g)"
"Mac_Snack_Wrap 4.4_oz_(125_g)"
"McFlurry_with_M&M'S_Candies_(12_fl_oz_cup) 10.9_oz_(310_g)"
"McRib_ 7.3_oz_(208_g)"
"Medium_French_Fries 4.1_oz_(117_g)"
"Mighty_Wings_(10_piece) 11.1_oz_(314_g)"
     # etc

;
### Nutrition Information

<table>
<thead>
<tr>
<th>Item</th>
<th>Calories</th>
<th>Calories from Fat</th>
<th>Fat (g)</th>
<th>Saturated Fat (g)</th>
<th>Cholesterol (mg)</th>
<th>Sodium (mg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacon Buffalo Ranch McChicken 5.6 oz (159 g)</td>
<td>420</td>
<td>180</td>
<td>20</td>
<td>4</td>
<td>50</td>
<td>1250</td>
</tr>
<tr>
<td>Big Breakfast (Large Size Biscuit) 10 oz (283 g)</td>
<td>800</td>
<td>470</td>
<td>52</td>
<td>18</td>
<td>555</td>
<td>1680</td>
</tr>
<tr>
<td>Big Mac 7.6 oz (215 g)</td>
<td>550</td>
<td>260</td>
<td>29</td>
<td>10</td>
<td>75</td>
<td>970</td>
</tr>
<tr>
<td>Chicken McNuggets (10 piece) 5.7 oz (162 g)</td>
<td>470</td>
<td>270</td>
<td>30</td>
<td>5</td>
<td>65</td>
<td>900</td>
</tr>
<tr>
<td>Coca-Cola Classic (Medium) 21 fl oz cup</td>
<td>200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Diet Coke (Medium) 21 fl oz cup</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Double Quarter Pounder with Cheese++ 10 oz (283 g)</td>
<td>750</td>
<td>380</td>
<td>43</td>
<td>19</td>
<td>160</td>
<td>1280</td>
</tr>
<tr>
<td>Egg McMuffin 4.8 oz (135 g)</td>
<td>290</td>
<td>110</td>
<td>12</td>
<td>5</td>
<td>260</td>
<td>740</td>
</tr>
<tr>
<td>Frappe Caramel (Medium) 16 fl oz cup</td>
<td>550</td>
<td>200</td>
<td>23</td>
<td>15</td>
<td>80</td>
<td>160</td>
</tr>
<tr>
<td>Hamburger 3.5 oz (100 g)</td>
<td>250</td>
<td>80</td>
<td>9</td>
<td>3</td>
<td>25</td>
<td>480</td>
</tr>
<tr>
<td>Hash Brown 2 oz (56 g)</td>
<td>150</td>
<td>80</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>310</td>
</tr>
<tr>
<td>Large French Fries 5.4 oz (154 g)</td>
<td>500</td>
<td>220</td>
<td>25</td>
<td>3</td>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>Mac Snack Wrap 4.4 oz (125 g)</td>
<td>330</td>
<td>170</td>
<td>19</td>
<td>7</td>
<td>45</td>
<td>670</td>
</tr>
<tr>
<td>McFlurry with M&amp;M's Candies (12 fl oz cup) 10.9 oz (310 g)</td>
<td>650</td>
<td>210</td>
<td>23</td>
<td>14</td>
<td>50</td>
<td>180</td>
</tr>
<tr>
<td>McRib 7.3 oz (208 g)</td>
<td>500</td>
<td>240</td>
<td>26</td>
<td>10</td>
<td>70</td>
<td>980</td>
</tr>
<tr>
<td>Medium French Fries 4.1 oz (117 g)</td>
<td>380</td>
<td>170</td>
<td>19</td>
<td>2</td>
<td>0</td>
<td>270</td>
</tr>
<tr>
<td>Mighty Wings (10 piece) 11.1 oz (314 g)</td>
<td>960</td>
<td>570</td>
<td>63</td>
<td>13</td>
<td>295</td>
<td>2900</td>
</tr>
</tbody>
</table>

### Carbohydrate, Protein, Vit A, Vit C, Calcium, Iron Information

<table>
<thead>
<tr>
<th>Item</th>
<th>Carbohydrates (g)</th>
<th>Protein (g)</th>
<th>Vit A (IU)</th>
<th>Vit C (mg)</th>
<th>Calcium (mg)</th>
<th>Iron (mg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacon Buffalo Ranch McChicken 5.6 oz (159 g)</td>
<td>41</td>
<td>20</td>
<td>2</td>
<td>10</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Big Breakfast (Large Size Biscuit) 10 oz (283 g)</td>
<td>56</td>
<td>28</td>
<td>15</td>
<td>2</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>Big Mac 7.6 oz (215 g)</td>
<td>46</td>
<td>25</td>
<td>4</td>
<td>2</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Chicken McNuggets (10 piece) 5.7 oz (162 g)</td>
<td>30</td>
<td>22</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Coca-Cola Classic (Medium) 21 fl oz cup</td>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Diet Coke (Medium) 21 fl oz cup</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Double Quarter Pounder with Cheese++ 10 oz (283 g)</td>
<td>42</td>
<td>48</td>
<td>10</td>
<td>2</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>Egg McMuffin 4.8 oz (135 g)</td>
<td>31</td>
<td>17</td>
<td>10</td>
<td>0</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>Frappe Caramel (Medium) 16 fl oz cup</td>
<td>79</td>
<td>9</td>
<td>20</td>
<td>0</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>Hamburger 3.5 oz (100 g)</td>
<td>31</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>Hash Brown 2 oz (56 g)</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Large French Fries 5.4 oz (154 g)</td>
<td>63</td>
<td>6</td>
<td>0</td>
<td>20</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Mac Snack Wrap 4.4 oz (125 g)</td>
<td>26</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>McFlurry with M&amp;M's Candies (12 fl oz cup) 10.9 oz (310 g)</td>
<td>96</td>
<td>13</td>
<td>15</td>
<td>0</td>
<td>45</td>
<td>8</td>
</tr>
<tr>
<td>McRib 7.3 oz (208 g)</td>
<td>44</td>
<td>22</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Medium French Fries 4.1 oz (117 g)</td>
<td>48</td>
<td>4</td>
<td>0</td>
<td>15</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Mighty Wings (10 piece) 11.1 oz (314 g)</td>
<td>40</td>
<td>60</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>15</td>
</tr>
</tbody>
</table>

# etc
solve;

printf {f in FOOD: Buy[f] > 0.3}: "%-60s %6.2f %4d \n", f, Buy[f], amt["Cal",f];
printf {i in NUTR}: "%-60s %7.1f (%4d)\n", i, sum {j in FOOD} amt[i,j] * Buy[j], nutr_ideal[i];

Complete AMPL model can be found here:


Here’s the python notebook for the McDonalds diet problem:

http://www.princeton.edu/~rvdb/307/python/IdealDiet.ipynb

Complete table of nutritional data can be found at:

rvdb@stars $ ampl idealDiet2014b.mod
LOQO 7.00: verbose=0
ignoring integrality of 296 variables
LOQO 7.00: optimal solution (14 QP iterations, 14 evaluations)
primal objective 2000
  dual objective 1999.999986

Chocolate_Chip_Cookie 1_cookie_(33_g)         1.63 160
Diet_Coke_(Child) 12_fl_oz_cup                0.38 0
Diet_Coke_(Medium) 21_fl_oz_cup               0.32 0
Diet_Coke_(Small) 16_fl_oz_cup                0.56 0
EQUAL_0_Calorie_Sweetener 1_pkg_(1.0_g)      21.32 0
Hotcakes 5.3_oz_(151_g)                       1.83 350
Iced_Tea_(Child) 12_fl_oz_cup                 1.45 0
Iced_Tea_(Large) 30_fl_oz_cup                 0.38 0
Iced_Tea_(Medium) 21_fl_oz                    0.56 0
Iced_Tea_(Small) 16_fl_oz_cup                 0.56 0
SPLENDA_No_Calorie_Sweetener 1_pkg_(1.0_g)    21.32 0
Side_Salad 3.1_oz_(87_g)                      1.97 20

Cal                         2000.0 (2500)
CalFat                      609.6 ( 600)
Fat                         68.2 (  65)
SatFat                      23.9 (  20)
Chol                        242.3 ( 300)
Sodium                     2870.7 (2400)
Carbo                       333.7 ( 300)
Protein                     59.9 (  50)
VitA                        118.8 ( 100)
VitC                        109.2 ( 100)
Calcium                    92.8 ( 100)
Iron                        80.1 ( 100)
subject to NotTooMuchSweetener:
    Buy["EQUAL_0_Calorie_Sweetener 1_pkg_(1.0_g)"]
    +Buy["SPLENDA_No_Calorie_Sweetener 1_pkg_(1.0_g)"]
    <= 2* ( 
        Buy["Iced_Tea_(Child) 12_fl_oz_cup"]
        +Buy["Iced_Tea_(Large) 30_fl_oz_cup"]
        +Buy["Iced_Tea_(Medium) 21_fl_oz"]
        +Buy["Iced_Tea_(Small) 16_fl_oz_cup"]
    );

Output:

Chocolate_Chip_Cookie 1_cookie_(33_g) 2.19 160
Diet_Coke_(Child) 12_fl_oz_cup 0.63 0
Diet_Coke_(Medium) 21_fl_oz_cup 0.44 0
Diet_Coke_(Small) 16_fl_oz_cup 0.94 0
EQUAL_0_Calorie_Sweetener 1_pkg_(1.0_g) 10.47 0
Fat_Free_Chocolate_Milk_Jug 1_carton_(236_ml) 0.68 130
Hotcakes 5.3_oz_(151_g) 1.53 350
Iced_Tea_(Child) 12_fl_oz_cup 7.24 0
Iced_Tea_(Large) 30_fl_oz_cup 1.02 0
Iced_Tea_(Medium) 21_fl_oz 1.98 0
Iced_Tea_(Small) 16_fl_oz_cup 1.98 0
SPLENDA_No_Calorie_Sweetener 1_pkg_(1.0_g) 10.47 0
Side_Salad 3.1_oz_(87_g) 1.83 20

Cal 2000.0 (2500)
CalFat 601.9 (600)
Fat 67.4 (65)
SatFat 23.9 (20)
Chol 243.5 (300)
Sodium 2742.7 (2400)
Carbo 313.7 (300)
Protein 59.9 (50)
VitA 118.4 (100)
VitC 110.0 (100)
Calcium 101.9 (100)
Iron 80.1 (100)
Third Run: Enforce Integrality Constraints

To do that we need a “Mixed Integer Linear Programming” solver such as Gurobi.

Output:

Chocolate_Chip_Cookie 1_cookie_(33_g) 6.00 160
Egg_McMuffin 4.8_oz_(135_g) 1.00 290
Fat_Free_Chocolate_Milk_Jug 1_carton_(236_ml) 2.00 130
Ketchup_Packet 1_pkg_(10_g) 5.00 10
Side_Salad 3.1_oz_(87_g) 1.00 20
Strawberry_Preserves 0.5_oz_(14_g) 12.00 35

Cal 2000.0 (2500)
CalFat 530.0 ( 600)
Fat 60.0 ( 65)
SatFat 23.0 ( 20)
Chol 330.0 ( 300)
Sodium 2060.0 (2400)
Carbo 330.0 ( 300)
Protein 48.0 ( 50)
VitA 97.0 ( 100)
VitC 83.0 ( 100)
Calcium 99.0 ( 100)
Iron 83.0 ( 100)
Linear Programming
**Standard Form.**

\[
\begin{align*}
\text{maximize} & \quad c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\
\text{subject to} & \quad a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n \leq b_1 \\
& \quad a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n \leq b_2 \\
& \quad \vdots \\
& \quad a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n \leq b_m \\
& \quad x_1, x_2, \ldots, x_n \geq 0.
\end{align*}
\]

**Why it’s hard:**

- Lots of variables (\(n\) of ’em).
- Lots of “boundaries” to check (the inequalities).

**Why it’s not impossible:**

- All expressions are linear.
Climate Change
Average Daily Temperatures at McGuire AFB


Avg. Temp. (F): 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

Temperature data from McGuire AFB, Data from NOAA, 55+ years.
Regression Models

Let $T_d$ denote the average temperature in degrees Fahrenheit on year $d \in D$ where $D$ is the set of years from 1955 to 2010.

$$T_d = x_0 + x_1 d + \varepsilon_d.$$  \hfill \text{linear trend}  

$$+ \varepsilon_d.$$  \hfill \text{“error” term}

The parameters $x_0$ and $x_1$ are unknown regression coefficients.

Either

$$\min \sum_{d \in D} |\varepsilon_d|$$  \hfill \text{Least Absolute Deviations (LAD)}

or

$$\min \sum_{d \in D} \varepsilon_d^2$$  \hfill \text{Least Squares}
Learn how to formulate optimization problems.

Learn to distinguish easy problems from hard ones from impossible ones.

Learn some of the theory of Linear Programming (Duality Theory!).

Learn how to express optimization problems in AMPL and solve them.