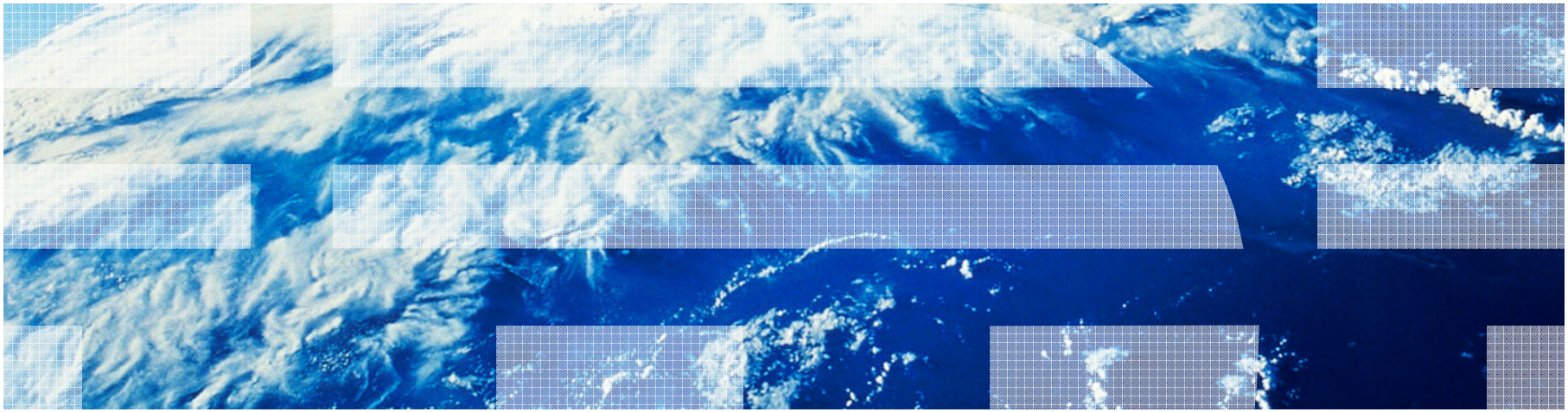
IBM Predictive Analytics Virtual Users' Group Meeting
March 30, 2016



Welcome to our March 2016 Virtual Users' Group

- **Welcome back to Past Attendees and Welcome to New Clients!**
  - Please continue to check out our blog for new information regarding these sessions as well as other interesting articles https://www.ibm.com/developerworks/mydeveloperworks/blogs/sca/?lang=en_us
- **Please write your questions to the Q&A Group via SmartCloud Meeting**
  - As we are expecting a lot of participants in today's session, all lines have been muted. Please send your questions via SmartCloud Meeting Chat:
    - This is the conversation text area in the lower left corner of the meeting sidebar
  - If you have further questions after the meeting please contact Kitte Knight at kknight@us.ibm.com
- **Our goal is to present topics of common interest in Optimization and create an open forum for our clients to exchange information between us and other clients**
- **If you should encounter any problems during the web conference please contact LOTUS LIVE HELP NUMBER: 1-888-376-0105**

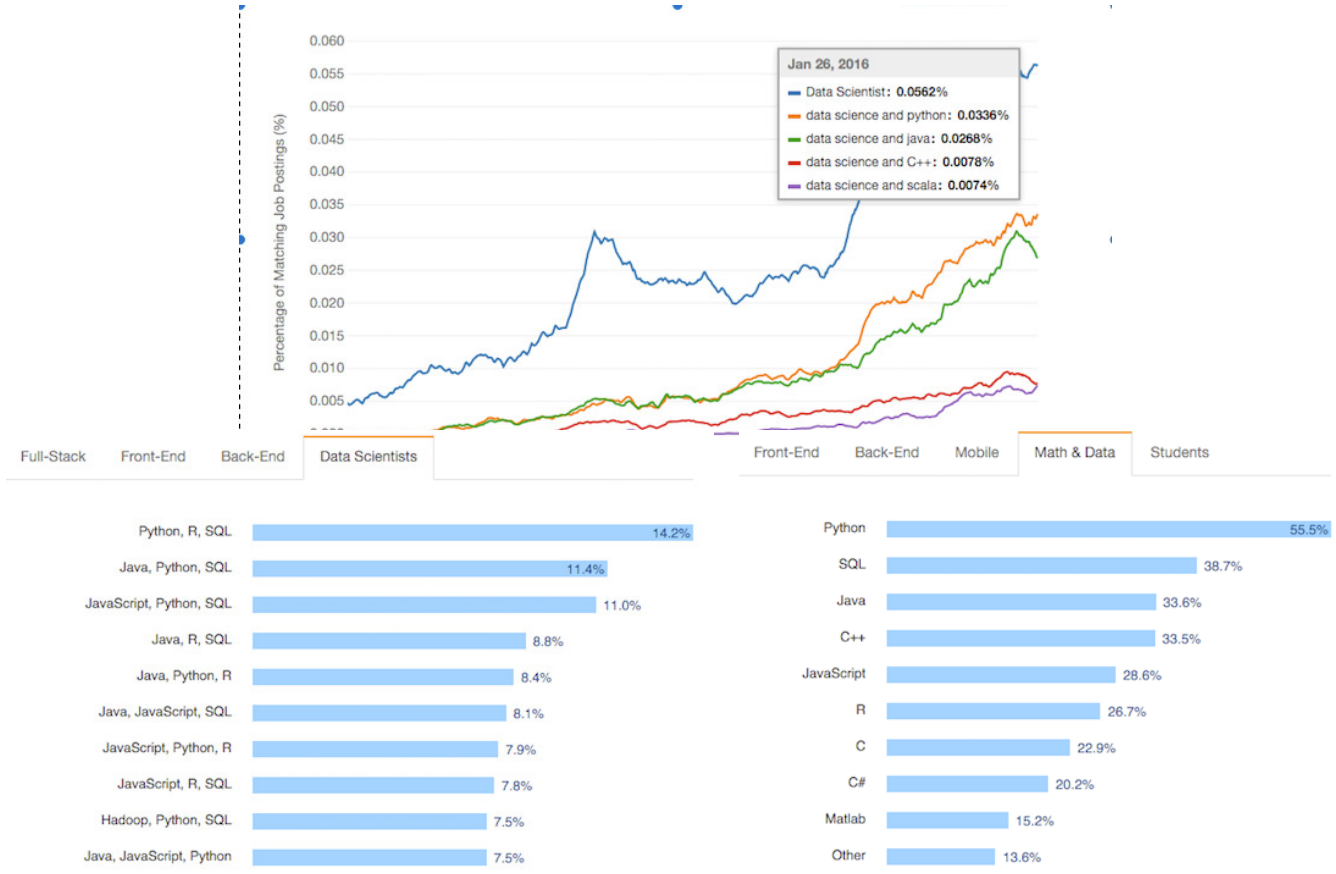
Speaker: **Vincent Beraudier**

Decision Optimization - Modeling Architect

OPL Language and IDE Development Manager
Python experience Development Manager

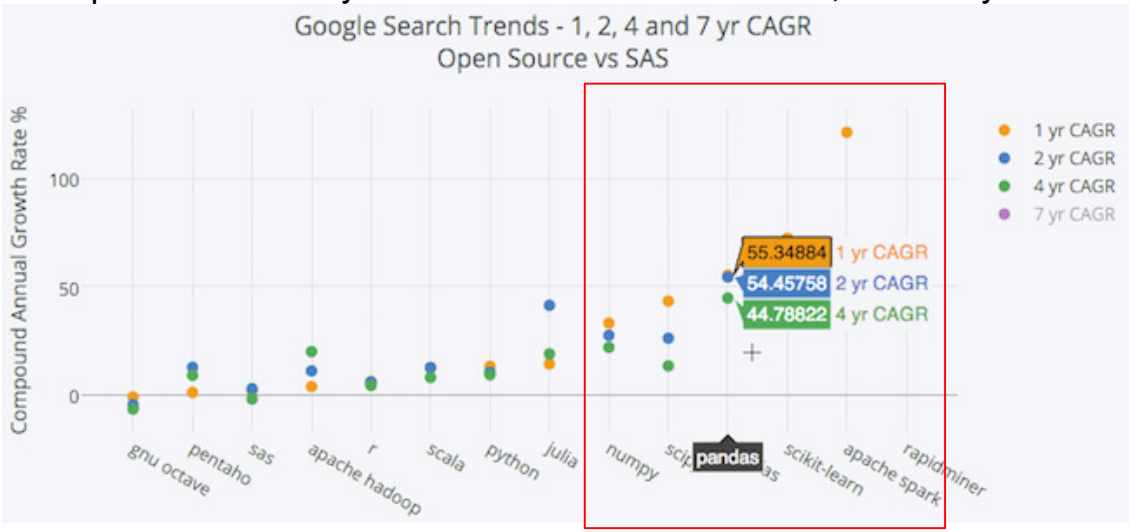
WHY PYTHON?

The Python wave in Data Science

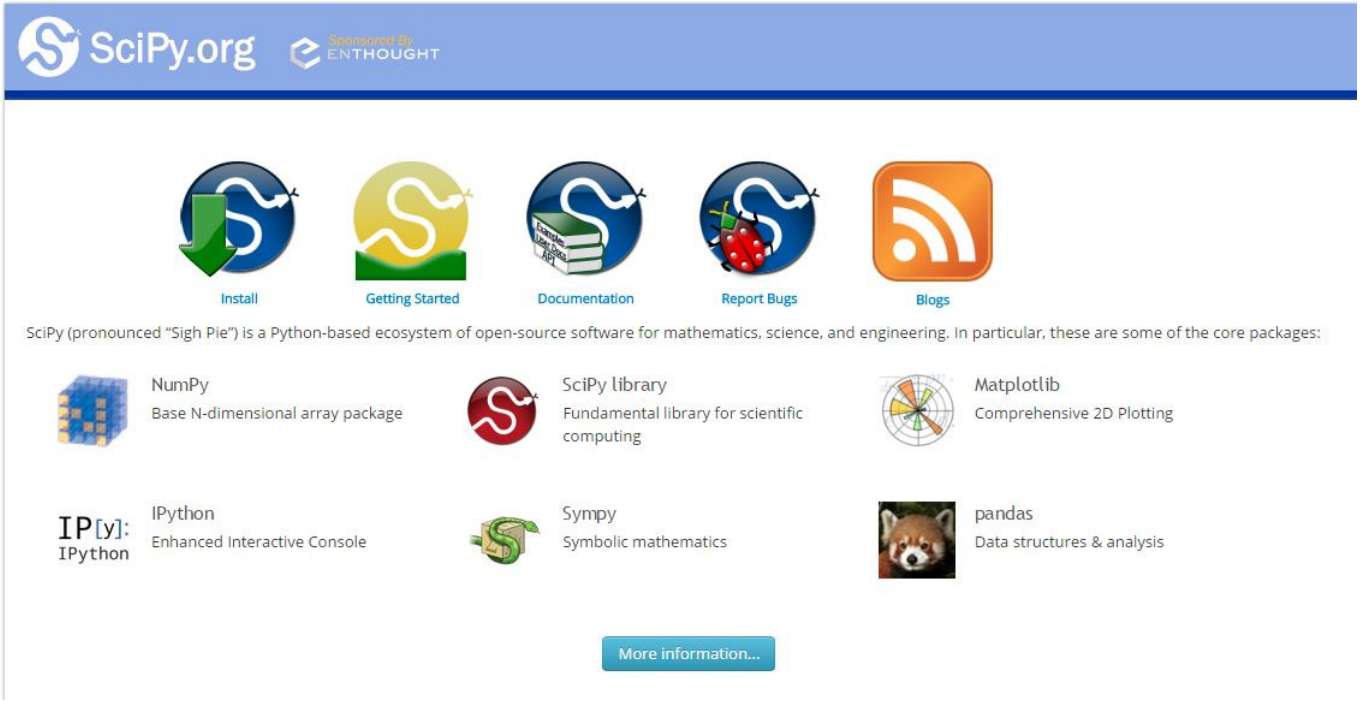


Data analytics and machine learning: spectacular growth

- Growth in Python search volume is not indicative of increased Python usage within data science teams.
- Search data for Pandas, Numpy, and scikit-learn, popular Python data analytics and machine learning add-on packages, more accurately reflects adoption.
- The growth has been spectacular: A 4 year CAGR for Pandas of 45%, and a 2 year CAGR for scikit-learn of 58%.



Python for scientific people



Scikits: complement SciPy with scientific computing routines.

SciKits

Home | About SciKits | Download a SciKit | Contribute

SciKits Index

scikit-aero	Aeronautical engineering calculations in Python.
scikit-allel	A Python package for exploring and analysing genetic variation data.
scikit-bio	Data structures, algorithms and educational resources for bioinformatics.
scikit-chainer	scikit-learn like interface for chainer
scikit-chem	A set of python modules for cheminformatics
scikit-commpy	Digital Communication Algorithms with Python
scikit-criteria	Multiple-criteria decision analysis package
scikit-cuda	Python interface to GPU-powered libraries
scikit-dsdp	Python interface to DSDP semidefinite programming library
scikit-fmm	An extension module implimenting the fast marching method
scikit-fusion	A Python module for data fusion built on top of factorized models.
scikit-fuzzy	Fuzzy logic toolkit for SciPy
scikit-geodesic	A SciPy tool for computing geodesics in an isotropic Riemannian manifold o
scikit-gpuppy	Gaussian Process Uncertainty Propagation with PYthon
scikit-image	Image processing routines for SciPy
scikit-learn	A set of python modules for machine learning and data mining
scikit-monaco	Python modules for Monte Carlo integration
scikit-multilearn	A set of python modules for multi-label classification
scikit-nano	Python toolkit for generating and analyzing nanostructure data
scikit-neuralnetwork	Neural Network wrapper for pylearn2 compatible with scikit-learn.
scikit-rf	Open Source RF Engineering
scikit-ribo	A scikit framework for joint analysis of Riboseq and RNAseq data.
scikit-spectra	Spectroscopy in Python built on Pandas.
scikit-tensor	Python module for multilinear algebra and tensor factorizations
scikit-tracker	Object detection and tracking for cell biology
scikit-umfpack	Python interface to UMFPACK sparse direct solver.
scikit-vi	Scikit providing Virtual Instruments
scikit-video	Video processing routines for SciPy
scikit-xray	Data analysis tools for X-ray science
mcts	A monte carlo tree search library

NOTEBOOKS

Notebooks: jupyter and zeppelin

- What do they say?
- <<The Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.>>
- <<interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media>>
- <<It aims to be an agile tool for both exploratory computation and data analysis, and provides a platform to support reproducible research**>>**

- <<The Notebook has support for multiple programming languages, including those popular in Data Science such as Python, R and Scala.>>
- <<Code can produce rich output such as images, videos, LaTeX, and JavaScript. Interactive widgets can be used to manipulate and visualize data in realtime.>>
- <<Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, dplyr, etc.>>

- **Notebooks are a strategic component of Analytic Platforms of IBM, Microsoft, Google and Amazon**

Aiming professional data scientists (as of today)

Apache Zeppelin (incubating)

A web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive and collaborative documents with SQL, Scala and more.

Watch the video | Get Zeppelin

Multi-purpose Notebook

The Notebook is the place for all your needs

- Data Ingestion
- Data Discovery
- Data Analytics
- Data Visualization & Collaboration

Multiple language backend

Zeppelin interpreter concept allows any language/data-processing-backend to be plugged into Zeppelin. Currently Zeppelin supports many interpreters such as Scala(with Apache Spark), Python(with Apache Spark), SparkSQL, Hive, Markdown and Shell.

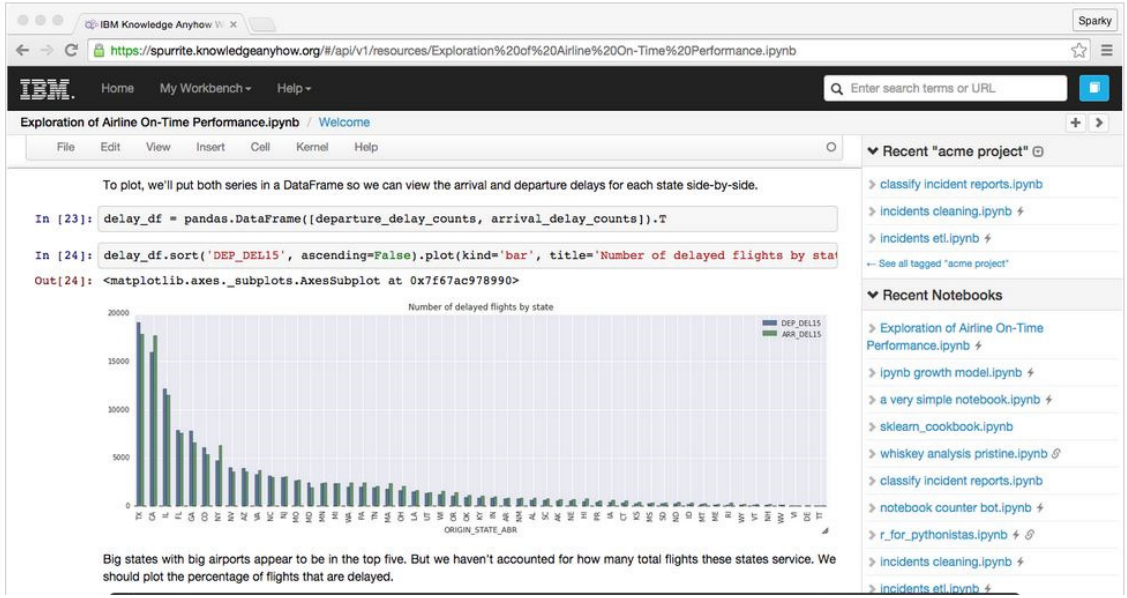
Aiming scientific community

IP[y]: IPython
Interactive Computing

Install · Docs · Videos · News · Cite · Sponsors · Donate

IPython provides a rich architecture for interactive computing with:

- Powerful interactive shells (terminal and Qt-based).
- A browser-based notebook with support for code, text, mathematical expressions, inline plots and other rich media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.
- Easy to use, high performance tools for parallel computing.



Execute code

Express the business constraints

First constraint: if the distance is suspect, it needs to be excluded from the problem.

In [15]:
```
for h in hubs:
    for p in points:
        if get_distance(h, p) >= BIGNUM:
            mdl.add_constraint(link_vars[h, p] == 0)
```

Second constraint: eliminates arcs with a distance that is null

In [16]:
```
for h in hubs:
    mdl.add_constraint(link_vars[h, h] == 0)
```

Third constraint: each point must be linked to a real hub

In [17]:
```
for p in points:
    for h in hubs:
        mdl.add_constraint(link_vars[h, p] <= hub_vars[h])
```

Fourth constraint: each non-hub point is linked to exactly one hub.

In [18]:
```
for p in points:
    if p in hubs:
        mdl.add_constraint(mdl.sum(link_vars[h, p] for h in hubs) == 1 - hub_vars[p])
    else:
        mdl.add_constraint(mdl.sum(link_vars[h, p] for h in hubs) == 1)
```

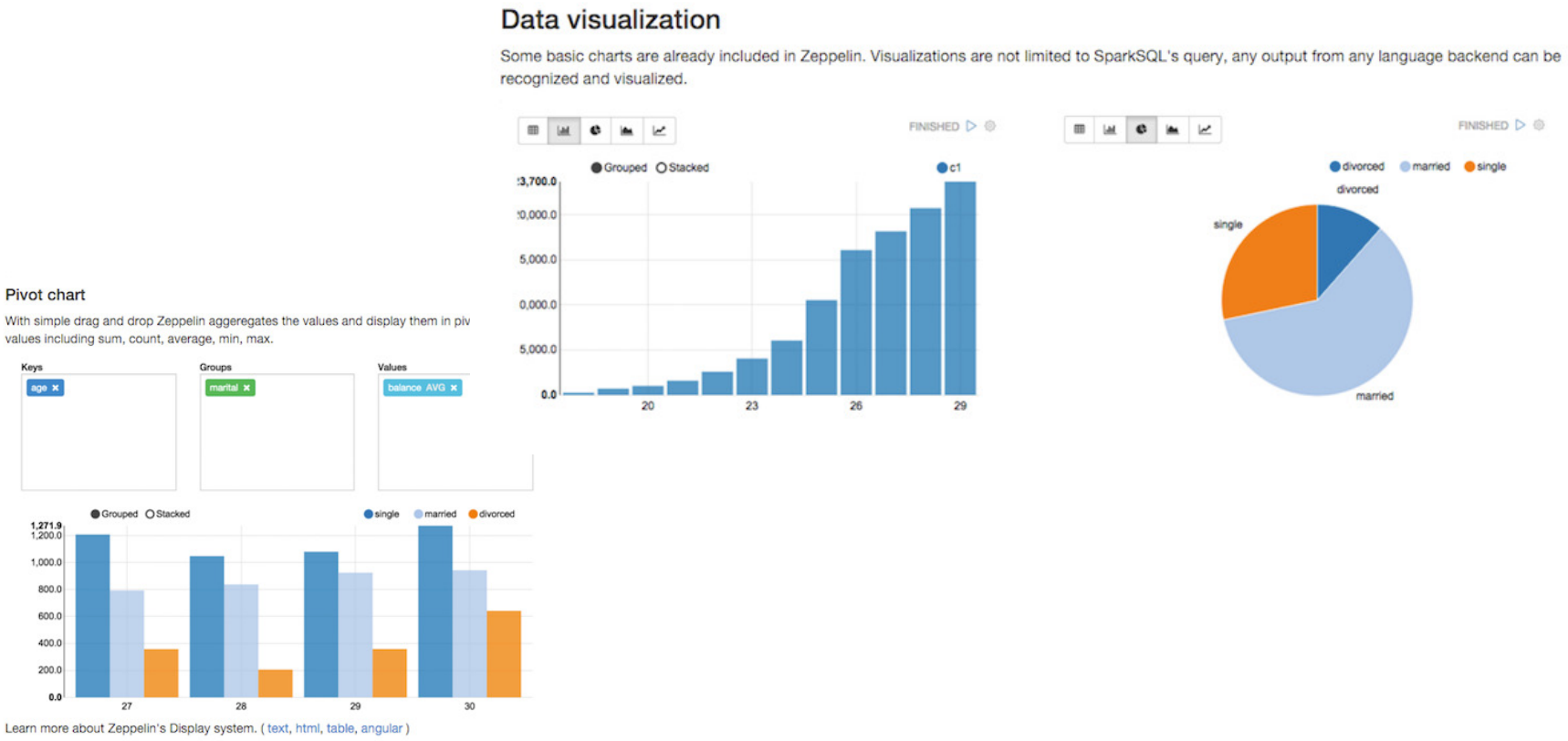
Fifth constraint: there is a fixed number of coffee shops to open

In [19]:
```
# total nb of open coffee shops
mdl.add_constraint(mdl.sum(hub_vars[h] for h in hubs) == nb_shops)

mdl.print_information()
```

```
Model: coffee shops
 - number of variables: 6480
   - binary=6480, integer=0, continuous=0
 - number of constraints: 6561
 -   LE=6400, EQ=161, GE=0, RNG=0
 - parameters: defaults
```

Data visualization



Online tutorials

Featured Notebooks

Learn techniques from the tutorials, samples, and recipes in these notebooks.

matplotlib Cookbook

Import

Learn to create and customize plots using these matplotlib recipes.

Last updated Dec 1, 2014

IBM Cloud Services Cookbook

Import

Connect to IBM cloud services such as IBM Cloudant and IBM Softlayer.

Last updated Jan 13, 2015

Scikit-Learn Cookbook

Import

Save time plotting ROC curves, building classification pipelines, doing hyperparameter optimization, etc. with these advanced scikit-learn recipes.

pandas Cookbook

Import

Learn to manipulate and analyze data using the popular *pandas* library.

Last updated Dec 10, 2014

Cloud Services Cookbook

Import

Connect to external cloud services such as Amazon S3, Dropbox, and Twitter.

Last updated Jan 20, 2015

R for Pythonistas

Import

If you know Python, pandas, and matplotlib, this notebook can get you started learning R equivalents such as data.frames and ggplot2.

Advanced Tutorials

Analyzing Financial Data

Tutorial - SoftLayer Object Storage

Import

Learn to access data in SoftLayer Object Storage.

Last updated Feb 19, 2015

Tutorial - Access DB2 Using Python

Import

Learn to access DB2 data using Python.

Last updated Feb 19, 2015

Tutorial - Intraday Seasonality Analysis Using Python

Import

Walk through an example of financial data analysis using Python.

Last updated Feb 19, 2015

Tutorial - Interactive Plots with IPython Widgets

Import

Learn to build interactive plots using IPython Widgets.

Last updated Feb 19, 2015

Tutorial - Analyzing Historical VSTOXX Data

Import

Learn to use data in SoftLayer Object Storage to perform analysis.

Last updated Feb 19, 2015

Tutorial - Access DB2 Using R

Import

Learn to access DB2 data using R.

Last updated Feb 19, 2015

Tutorial - Intraday Seasonality Analysis Using R

Import

Walk through an example of financial data analysis using R.

Last updated Feb 19, 2015

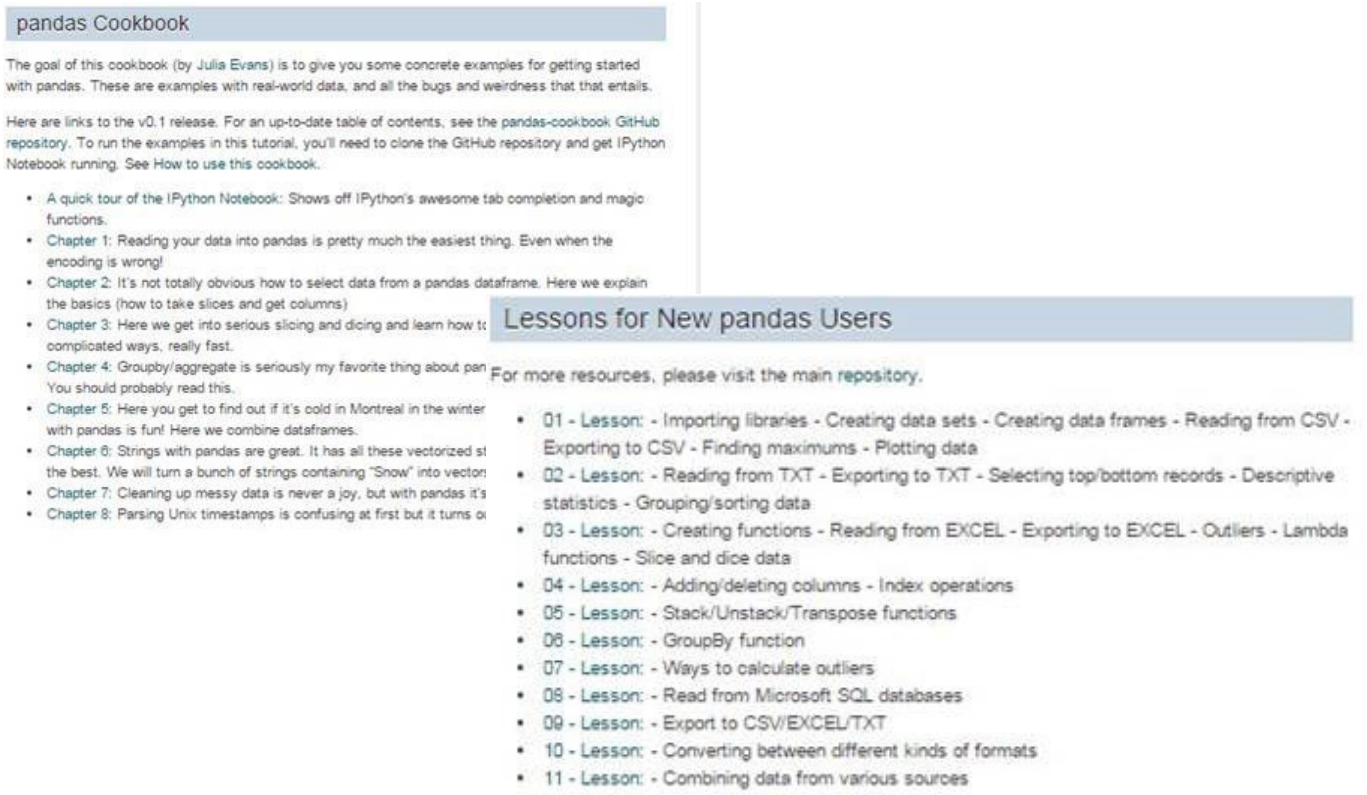
Financial Analysis with Pandas Cookbook

Import

Learn to load and analyze financial data from external sources.

Last updated Feb 19, 2015

Extensive support for learning



pandas Cookbook

The goal of this cookbook (by Julia Evans) is to give you some concrete examples for getting started with pandas. These are examples with real-world data, and all the bugs and weirdness that that entails.

Here are links to the v0.1 release. For an up-to-date table of contents, see the pandas-cookbook GitHub repository. To run the examples in this tutorial, you'll need to clone the GitHub repository and get IPython Notebook running. See How to use this cookbook.

- A quick tour of the IPython Notebook: Shows off IPython's awesome tab completion and magic functions.
- Chapter 1: Reading your data into pandas is pretty much the easiest thing. Even when the encoding is wrong!
- Chapter 2: It's not totally obvious how to select data from a pandas dataframe. Here we explain the basics (how to take slices and get columns)
- Chapter 3: Here we get into serious slicing and dicing and learn how t
complicated ways, really fast.
- Chapter 4: Groupby/aggregate is seriously my favorite thing about pan
You should probably read this.
- Chapter 5: Here you get to find out if it's cold in Montreal in the winter
with pandas is fun! Here we combine dataframes.
- Chapter 6: Strings with pandas are great. It has all these vectorized st
the best. We will turn a bunch of strings containing "Snow" into vector
- Chapter 7: Cleaning up messy data is never a joy, but with pandas it's
- Chapter 8: Parsing Unix timestamps is confusing at first but it turns o

Lessons for New pandas Users

For more resources, please visit the main repository.

- 01 - Lesson: - Importing libraries - Creating data sets - Creating data frames - Reading from CSV - Exporting to CSV - Finding maximums - Plotting data
- 02 - Lesson: - Reading from TXT - Exporting to TXT - Selecting top/bottom records - Descriptive statistics - Grouping/sorting data
- 03 - Lesson: - Creating functions - Reading from EXCEL - Exporting to EXCEL - Outliers - Lambda functions - Slice and dice data
- 04 - Lesson: - Adding/deleting columns - Index operations
- 05 - Lesson: - Stack/Unstack/Transpose functions
- 06 - Lesson: - GroupBy function
- 07 - Lesson: - Ways to calculate outliers
- 08 - Lesson: - Read from Microsoft SQL databases
- 09 - Lesson: - Export to CSV/EXCEL/TXT
- 10 - Lesson: - Converting between different kinds of formats
- 11 - Lesson: - Combining data from various sources

Report/blog/publication ready

Table of contents

Mathematical problem

We address the initial-value problem

$$u'(t) = -au(t), \quad t \in (0,T], \tag{1}$$
$$u(0) = I, \tag{2}$$

where a, I, and T are prescribed parameters, and $u(t)$ is the unknown function to be estimated. This mathematical model is relevant for physical phenomena featuring exponential decay in time.

Numerical solution method

We introduce a mesh in time with points $0 = t_0 < t_1 \cdots < t_N = T$. For simplicity, we assume constant spacing Δt between the mesh points: $\Delta t = t_n - t_{n-1}$, $n = 1, \ldots, N$. Let u^n be the numerical approximation to the exact solution at t_n.

The θ-rule is used to solve (1) numerically:

$$u^{n+1} = \frac{1-(1-\theta)a\Delta t}{1+\theta a\Delta t}u^n,$$

for $n = 0, 1, \ldots, N-1$. This scheme corresponds to

- The Forward Euler scheme when $\theta = 0$
- The Backward Euler scheme when $\theta = 1$
- The Crank-Nicolson scheme when $\theta = 1/2$

Implementation

The numerical method is implemented in a Python function `solver` (found in the `decay_mod` module):

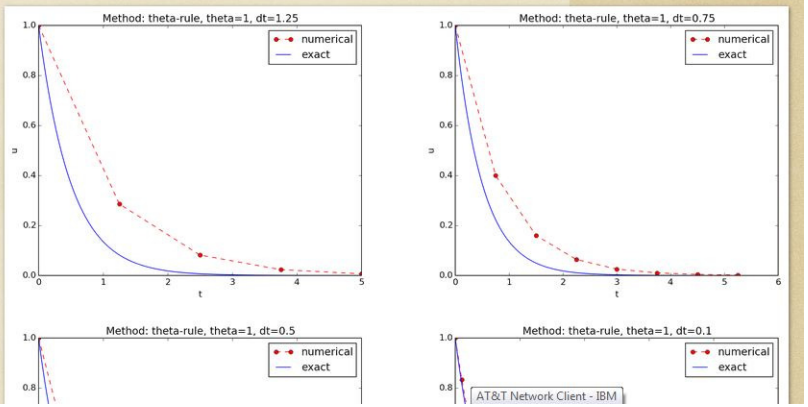
```
def solver(I, a, T, dt, theta):
    """Solve u'=-a*u, u(0)=I, for t in (0,T] with steps of dt."""
    dt = float(dt)            # avoid integer division
    N = int(round(T/dt))      # no of time intervals
    T = N*dt                  # adjust T to fit time step dt
    u = zeros(N+1)            # array of u[n] values
    t = linspace(0, T, N+1)   # time mesh

    u[0] = I                  # assign initial condition
    for n in range(0, N):     # n=0,1,...,N-1
        u[n+1] = (1 - (1-theta)*a*dt)/(1 + theta*dt*a)*u[n]
    return u, t
```

Numerical experiments

We define a set of numerical experiments where I, a, and T are fixed, while Δt and θ are varied. In particular, $I = 1$, $a = 2$, $\Delta t = 1.25, 0.75, 0.5, 0.1$.

The Backward Euler method



Conclusion

- Python is one of the most relevant tools to easily turn an idea into working code when dealing data-wrangling problems, and then visualize their results.

- It becomes the best one as soon as you look to deliver commercial software that uses data science or turn a model into production. Out of the box, here are some of its numerous advantages:
  - Interpreted, dynamically-typed language with a precise, efficient syntax.
  - Growing scientific packages: numerical and data analysis, plotting functionality, machine learning, scientific computing …
  - Jupyter/Zeppelin notebooks to try-out code snippets and create business reports.
  - Large community to get help and open source packages
  - Common programming language used all across different teams in the organization.
  - Data ingest is really really simple

OPTIMIZATION FOR PYTHON

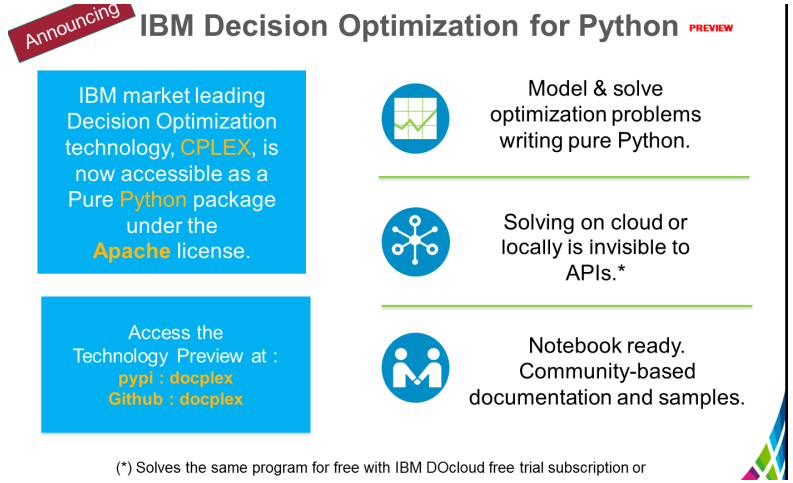
Optimization modeling and Python?

- Take advantage of python expressiveness (generators, aggregators, operator overloading, tuples…).
- 1 single language to create the constraints AND do the workflow.
- Ecosystem, ease of use, proven robustness, data ingest

- A programming language with the flavor of a scripting language
  Work flow and mathematical description are part of the language, no memory management

- Active open source community (**>70k** projects)
- 1st class citizen in web service apis economy
- Easily integrate with other languages (notebooks)
- Wondering how to do something? Someone already did it for you…

IBM Decision Optimization for Python



- **Easily formulate your optimization models and solve them with DOcloud solve service or CPLEX local solver (with 0 code change).**
- **Access to free solve capabilities to discover this new API is made easy thanks to our cloud free trial and our new CPLEX Optimization Studio free Community Edition (aka COS CE): you can get access to any of those two with the help of one mail address.**
- **Available through the standard Python pip install with no need to download anything else or contact any IBM person if you go full cloud.**
- **Just look for docplex in your browser to get access to docplex pypi repo or doc.**

Simple effective documentations

Discovering the IBM Decision Optimization technologies

If you are new to optimization technologies, these topics present an overview of the algorithms, their specific application domains, and a list of books and free online trainings.

- *Overview of mathematical programming*
- *Overview of constraint programming*
- *Mathematical programming versus constraint programming*

Getting started with DOcplex

- Setting up a Python environment
- Setting up an optimization engine
- Support for DOcplex

Developing with DOcplex

Mathematical Programming Modeling for Python using docplex.mp (DOcplex.MP)

- Examples of mathematical programming
- Creating a MP model in a nutshell
- Mathematical programming reference manual

Constraint Programming Modeling for Python using docplex.cp (DOcplex.CP)

- Examples of constraint programming
- Creating a CP model in a nutshell
- Constraint programming reference manual

New to Python? It's not a problem.

Setting up a Python environment ¶

To get started using the Mathematical Programming Modeling for Python feature of IBM® Decision Optimization CPLEX® Modeling for Python library (DOcplex.MP), you first need to verify that your system meets the requirements and install Python and DOcplex. As an alternative, you can choose to use the Data Scientist Workbench and not need to install anything on your computer.

System requirements

- **Python with version >= to 2.7.9 or 3.4.x** on 64-bit operating systems: Available from Python.org.
  - This library is not supported on Python 3.5 as the CPLEX engine is not available for this version yet.
  - Python version lifespans: here
- Only **Windows**, **Mac** and **Linux** operating systems are supported. Your computer must support SSL.
- See the section on *Troubleshooting* for help diagnosing installation and execution problems.

Get Python development tools

- If you are new to Python, you might want a development studio with editors and debuggers. Both PyCharm and PyDev have free editions.
- You can download Python 2.7.9 or 3.4.2 from Python.org.
- A good alternative is to use Anaconda, which includes a Python interpreter and over 195 of the most popular Python packages for science, math, engineering, and data analysis.
- You can also turn a Visual Studio into a Python editor. See Python Tools for Visual Studio project from Microsoft.

Installing the CPLEX modeling library

The IBM Decision Optimization CPLEX Modeling for Python (DOcplex) library can be installed via *pip* from PyPI.

Use *pip* to install the modeling library:

More resources on Python

- Free online trainings
  - 5-course series, the Python for Everybody: develop programs to gather, clean, analyze, and visualize data, by the Michigan University
  - An Introduction to Interactive Programming in Python - Part 1 from RICE University
  - An Introduction to Interactive Programming in Python - Part 2 from RICE University
  - Exploring CO2 Emissions Data using Pandas data frames in Python
  - Exploring and Visualizing the Number of Out-of-School Children Around the World using Python
  - Introduction to Python for Data Science
- Books
  - Learn Python the Hard Way -
  - Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython - Wes McKinney
  - Introducing Python: Modern Computing in Simple Packages - Bill Lubanovic
  - Learning Python, 5th Edition - Mark Lutz
- Scientific Python Conferences
  - SciPy Conferences

DEMO

Notebook for Optimization

Finding optimal locations of new stores

This tutorial includes everything you need to setup the IBM Decision Optimization CPLEX Modeling for Python (DOcplex) build a Mathematical Programming model and get its solution by solving the model on the cloud with IBM ILOG CPLEX Optimizer.

When you finish this tutorial, you'll have a foundational knowledge of *Prescriptive Analytics*.

> This notebook is part of the **Prescriptive Analytics for Python**
>
> It requires a valid subscription to **Decision Optimization on the Cloud**. Try it for free here Table of contents:

Table of contents:

Business problem

- A fictional Coffee Company plans to open N shops in the near future and needs to determine where they should be located knowing the following:
  - Most of the customers of this coffee brewer enjoy reading and buying books, so the goal is to locate those shops in such a way that all the city bookstores are within minimal walking distance.
- We will use Chicago open data for this example.
- We will implement a K-Median model to get the optimal location of our future shops.

Installing docplex and put your credentials.

Use decision optimization

Step 1: Download the library

Run the following code to install Decision Optimization CPLEX Modeling library. The *DOcplex* library contains the two modeling packages, Mathematical Programming and Constraint Programming, referred to earlier.

In [80]:
```
!pip install docplex -q
```

Step 2: Set up the prescriptive engine

- Subscribe to the Decision Optimization on Cloud solve service.
- Get the service URL and your personal API key.

In [81]:
```
from docplex.mp.context import *

url= "https://api-oaas.docloud.ibmcloud.com/job_manager/rest/v1/"
key = "PUT_YOUR_KEY_HERE"

ctx = Context.make_default_context(url=url, key=key)
```

Downloading and parsing json is just easy

Declare the list of bookstores

Parse the json file to get the list of bookstores and store them as Python elements.

In [85]:
```
def build_cities_from_url(url, name_pos, lat_long_pos):
    import requests
    import json

    r = requests.get(url)
    myjson = json.loads(r.text, parse_constant='utf-8')
    myjson = myjson['data']

    cities = []
    k = 1
    for location in myjson:
        uname = location[name_pos]
        try:
            latitude = float(location[lat_long_pos][1])
            longitude = float(location[lat_long_pos][2])
        except TypeError:
            latitude = longitude = None
        try:
            name = str(uname)
        except:
            name = "???"
        name = "P_%s_%d" % (name, k)
        if latitude and longitude:
            cp = NamedPoint(name, longitude, latitude)
            cities.append(cp)
            k += 1
    return cities
```

In [86]:
```
hubs = build_cities_from_url('https://data.cityofchicago.org/api/views/x8fc-8rcq/rows.json?accessType=DOWNLOAD',
                             name_pos=12,
                             lat_long_pos=18)
```

In [87]:
```
print("There are %d bookstores in Chicago" % (len(hubs)))
```

```
There are 80 bookstores in Chicago
```

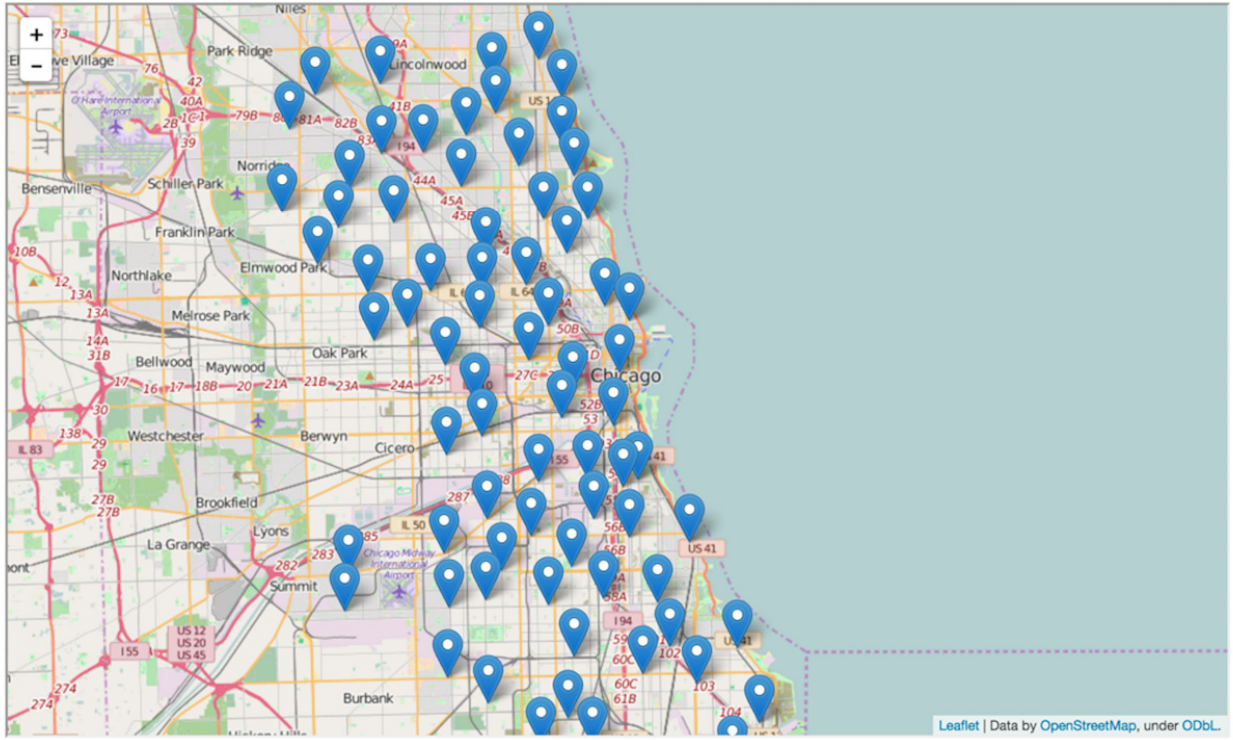
Visualizing the input data

```
In [10]: !pip install folium
```

```
Downloading/unpacking folium
  Downloading folium-0.2.0.tar.gz (67kB): 67kB downloaded
  Running setup.py (path:/home/notebook/ka_env/build/folium/setup.py) egg_info for package folium
Successfully installed folium
Cleaning up...
```

```
In [11]: import folium
map_osm = folium.Map(location=[41.878, -87.629], zoom_start=11)
for hub in hubs:
    lt = hub.y
    lg = hub.x
    folium.Marker([lt, lg]).add_to(map_osm)
map_osm
```

Out[11]:



Step 5: Set up the prescriptive model

In [91]:
```
from docplex.mp.environment import Environment
env = Environment()
env.print_information()
```

```
* system is: Linux 64bit
* Python is present, version is 2.7.11
* docplex is present, version is (1, 0, 455)
* CPLEX wrapper is not available
```

Create DOcplex model

It will contain all the business constraints and define the objective.

In [92]:
```
from docplex.mp.model import Model

mdl = Model("coffee shops", context=ctx)
```

```
Warning: CPLEX DLL not found, will solve on DOcloud
```

Define the decision variables

In [93]:
```
BIGNUM = 999999999

# ensure unique points
points = set(hubs)
#any point can be selected as a hub.
hubs = points

# vars
hub_vars = mdl.binary_var_dict(hubs, name="is_hub")
link_vars = mdl.binary_var_matrix(hubs, points, "link")
```

Express the business constraints

First constraint: if the distance is suspect, it needs to be excluded from the problem.

In [94]:
```
for h in hubs:
    for p in points:
        if get_distance(h, p) >= BIGNUM:
            mdl.add_constraint(link_vars[h, p] == 0, "ct_forbid_{0!s}_{1!s}".format(h, p))
```

Second constraint: eliminates arcs with a distance that is null

In [95]:
```
for h in hubs:
    mdl.add_constraint(link_vars[h, h] == 0)
```

Third constraint: each point must be linked to a real hub

In [96]:
```
for p in points:
    for h in hubs:
        mdl.add_constraint(link_vars[h, p] <= hub_vars[h])
```

Fourth constraint: each non-hub point is linked to exactly one hub.

In [97]:
```
for p in points:
    if p in hubs:
        mdl.add_constraint(mdl.sum(link_vars[h, p] for h in hubs) == 1 - hub_vars[p])
    else:
        mdl.add_constraint(mdl.sum(link_vars[h, p] for h in hubs) == 1)
```

Fifth constraint: there is a fixed number of coffee shops to open

In [98]:
```
# total nb of open coffee shops
mdl.add_constraint(mdl.sum(hub_vars[h] for h in hubs) == nb_shops)

mdl.print_information()
```

```
Model: coffee shops
 - number of variables: 6480
   - binary=6480, integer=0, continuous=0
 - number of constraints: 6561
 -   LE=6400, EQ=161, GE=0, RNG=0
 - parameters: defaults
```

Solve with Decision Optimization solve service

Solve the model on the cloud. As the model was created with a context that contained the URL and key of the solve service, we do not need to specify the context, nor the url or key when call solve().

In [100]:
```
print("#points=%d" % len(points))
print("#coffee shops  =%d" % nb_shops)

ok = mdl.solve(context=ctx)
if not ok:
    print("!!! Solve of the model fails")
```

```
#points=80
#coffee shops  =5
```

Step 6: Investigate the solution and then run an example analysis

The solution can be analyzed by displaying the location of the coffee shops on a map.

In [101]:
```
total_distance = mdl.objective_value
open_hubs = [h for h in hubs if float(hub_vars[h]) >= 0.9]
not_hubs = [h for h in hubs if h not in open_hubs]
edges = [(h, p) for p in points for h in hubs if float(link_vars[h, p]) >= 0.9]

print("total distance=%g" % total_distance)
print("#hubs={0}".format(len(open_hubs)))
for h in open_hubs:
    print("new coffee shop: {0!s}".format(h))
```

```
total distance=209.812
#hubs=5
new coffee shop: P_4455 N. Lincoln Avenue_65
new coffee shop: P_6100 W. Irving Park Road_5
new coffee shop: P_2111 W. 47th Street_7
new coffee shop: P_6 S. Hoyne Avenue_46
new coffee shop: P_9525 S. Halsted Street_79
```

In [29]:
```
import folium
map_osm = folium.Map(location=[41.878, -87.629], zoom_start=11)

for hub in hubs:
    lt = hub.y
    lg = hub.x
    folium.Marker([lt, lg]).add_to(map_osm)


for (p, h) in edges:
    coordinates = [[p.y, p.x], [h.y, h.x]]
    map_osm.line(coordinates, line_color='#FF0000', line_weight=5)

map_osm
```

Out[29]: