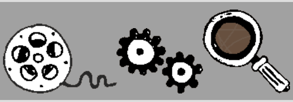
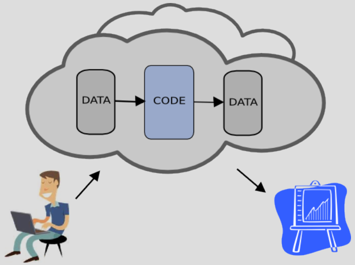


Mathpak

A platform for collaborative analytic apps

Bay Area R Users Group
10/02/2013

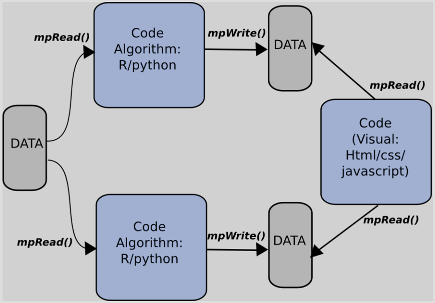
Mathpak



- A cloud based platform for developers to build and deploy analytic apps
  - Developers upload components to platform
    - Component = Code + Input/Output data
    - Component = Algorithm or Visualization
  - Developers compose apps by linking components in a pipeline
  - Apps executed on platform

How it works

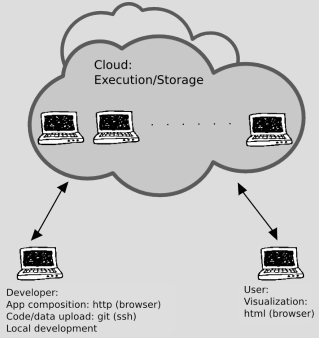
- Building blocks: Code and data
  - Code: Algorithms: R/python, Visuals: html/css/javascript)
  - Data: csv for (R), flat files for python/javascript
  - Code reads input data, write output data



- Components use platform library for data input/output
  - API to open/close/read/write data files
  - Data *statically bound* to code during app composition
- Pipeline: App composed by linking components
  - Pipeline can have several stages
  - Stage can have multiple components
  - Web interface used for linking

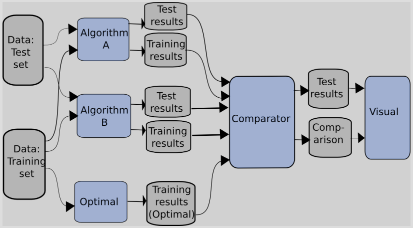
How it works(2)

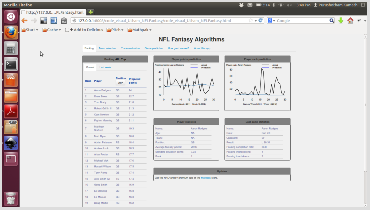
- Developer platform
  - Scalable cloud based system for data, code store and code execute
    - Platform executes app code pipeline end to end (all code components)
    - Platform generates algorithm output data for visuals
  - Environment for collaboration: wiki, message boards, IRC
  - Code/data can be private/public
  - Licensing: Code can use a proprietary license or GPL/BSD license



- Marketplace
  - Apps/data available on marketplace, other platforms (Facebook, API hubs)
  - Revenue: Split among platform and developers according to public algorithm

Example: Sports prediction





NFL Fantasy Algorithms

Rank	Player	Position All	Projected points
1	Aaron Rodgers	QB	24
2	Drew Brees	QB	22.7
3	Tom Brady	QB	21.6
4	Robert Griffin III	QB	21.3
5	Cam Newton	QB	21.2
6	Peyton Manning	QB	21.1
7	Matthew Stafford	QB	19.3
8	Matt Ryan	QB	18.6
9	Adrian Peterson	RB	18.4
10	Andrew Luck	QB	18.3
11	Arian Foster	RB	17.7
12	Michael Vick	QB	17.6
13	Russell Wilson	QB	17.5
14	Tony Romo	QB	17.4
15	Alex Smith (2)	TE	17.4
16	Geno Smith	QB	16.9
17	Eli Manning	QB	16.8
18	EJ Manuel	QB	16.3
19	Doug Martin	RB	16.2

Player statistics

Name:	Aaron Rodgers
Age:	NA
Team:	NA
Position:	QB
Average fantasy points:	20.08
Standard deviation points:	7.34
Rank:	1

Last game statistics

Name:	Aaron Rodgers
Date:	Sun 9/8
Opponent:	SF
Result:	L 28-34
Passing completion rate:	56.8
Passing interceptions:	1
Passing touchdowns	3

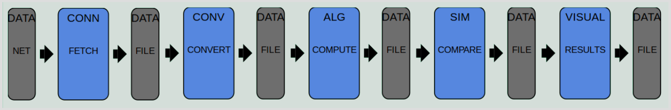
Updates

Get the NFLFantasy premium app at the Mathpak store.

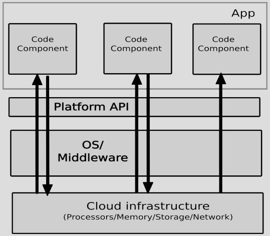
- Fantasy ranking of NFL players, game prediction
  - Goal: Predict fantasy ranking, game outcome based on game/player stats
- Building blocks for game prediction:
  - Algs: Logistic regression in R for game prediction
  - Comparator: Python:Compare algorithms/Vegas/other sources
  - Visual: html/css/javascript (d3.js, jquery, jstat, twitter-bootstrap)
- App deployed on platform, data available via API
  - http://cs0.mathpak.com/NFLPerfAnalyzer
  - Easy to extend
  - Add new ranking algorithms (supervised learning?)
  - Add better visualizations

Is this useful?

- Common pipeline for analysis
  - Components based: Plug and play with analytic components
  - Collaborative: Developers focus on their areas of expertise (algs/visuals)
  - Build within a framework or independent of framework
  - Leverage open source libraries (performance analytics, nltk, sci-kit, et. al.)



- Separates analytics from system design
  - Developer can focus on algorithms/data/visualizations
  - Avoid issues of system architecture, deployment, user acquisition, marketing



Why build (yet another) code/data hub?

- What you can do today:
    - Collaborate around code (github)
    - Build an analytic app for an app market
    - Publish on the Web (shiny for R)
    - Deploy an app (Heroku/Facebook)
    - Use algorithms as services
    - Build a custom solution on your own and publish API (mashape)

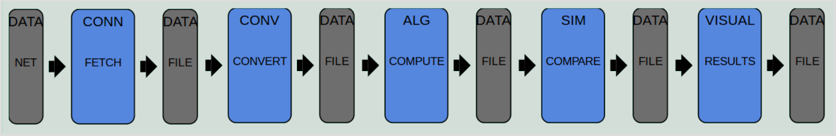
- So, do we really need another code/data hub?

- What you can't do (easily) today:
    - Collaborate around common frameworks for comparison
    - Mix/match programming languages/libraries, avoid proprietary/closed source platform/tools
    - Reuse components
    - Deploy analytic apps easily

What it is not (Related work)

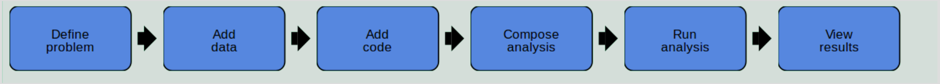
Category	Examples	Use
Code hubs	github, Rproject, Rforge, CRAN, Source	Storage,source code/project management
Data hubs	data.io, data.gov	Storage, repository, sharing
Deployment environments	Heroku	Execution
Development environments	RSource, RStudio/RPubs/Shiny	Development, sharing
Communities	Kaggle	Competitive data science
Platforms	Knime, Datameer, Platfora	Analytics
Services	algorithms.io, bigml, wise.io	Algorithms as a service
Tools	Google charts, Google Fusion Tables, Revolution, Tableau	Analysis
Marketplaces	mashery, mashape	Sell/use APIs

Technology



- Component names/handles
  - All code/data on the system gets a unique system wide "name"
  - Code use "handles" to read/write input/output data
- Apps created by building a pipeline of components
  - Developer links handles to data during app composition
  - Platform library maps "handles" to "names": Static binding
  - +ve: Mix and match components from different users
  - -ve: Needs careful planning of components, data, data formats
- Components types
  - Connectors: Download data from external source (Network to File)
  - Converters: Convert data from one format to another (File to File)
  - Algorithms: Process data (File to File)
  - Visualizations: Visualize data (File to Visual)

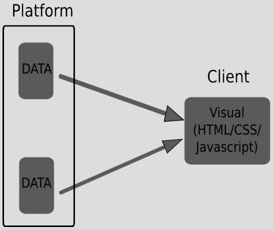
Development



- Create code/data
    - Developers *define* components (code/data) on the developer site
    - Generates a repo for each component (code/data)
    - Generates a system wide unique name
- Compose app from code and data
    - Generates an app definition xml file (used by platform library to find data)
- Local development:
    - Download/upload app components (skeleton code/data files) git clone/pull
    - *Add* data, code - use platform library API to read/write data
        - Library: File/chunking read/write, Single threaded, Local file I/O
        - Same library for local development/platform execution
    - Components/apps can be executed locally by developers
    - Upload code/data: git add/commit/push (git-annex for "larger" data)

Deployment

- Developer Platform
  - Executes app periodically or if input data changes
  - Makes code/data available on platform
  - Makes visualization available on marketplace/other platforms
  - Makes data available via an API



- Marketplace
  - Apps (end visualization) available for customers, including source code (subject to license/privacy)
- Revenue distribution
  - Basic algorithm: Traverse app pipeline tree, split revenue among components/platform

Performance

- Scale
    - System scales with increase in computation (number of cycles per app, number of apps)
    - System scales with increase in data, subject to data size limits
        - Limited by direct attached storage (local file system size)
    - Platform library being extended to scale for big data
        - R/python through Hadoop/other scaling technologies

More examples

- News trends
  - Identify news stories most discussed on social media
  - Correlates n-grams from news media RSS feeds with twitter data to discover most talked about news stories
  - Building blocks:
    - Conn: python: Download RSS streams from CNN, NYTimes
    - Alg: python: Finds most popular bigrams in news headlines (nltk)
    - Conn: python: Downloads recent tweets based on search for bigrams
    - Alg: python: Calculates tweet rate from twitter data, ranks topics
    - Visual: html/css/d3: Shows graphs

Status

- Current status
  - Developer site functional (with storage limits: 10 GB aggregate, 1GB per file)
  - Marketplace functional
  - Platform libraries (0.1.7.3) available on github/developer.mathpak.com
- Private alpha: In progress
  - Closed marketplace, closed developer site
  - Looking for developers to build apps
  - Hackathons, meetups over the next few weeks
  - Mail me at utham@mathpak.com,
    - if you have an interesting idea for an app/component
    - if you are interested in building apps/components (in R/python/html)
- Public alpha: Q4, 2013
  - Open marketplace, open developer site
- Development plan
  - Several new platform features under development
  - Platform API Ver 0.2.0.0 planned for early Q1 2014