# Metashift

Matthew D'Souza & Dongyu Zheng

August 30, 2015

## Contents

## 1 Abstract

Created for the League of Legends API Challenge 2.0, Metashift is a web app which uses Riot Games' API to query the game information and present a graphical analysis of thousands of League of Legends matches before and after the 5.13 patch. Many of the changes were made to the items with respect to the Ability Power stat. The goal of this project is to systematically look for a shift in the 'metagame' - the most popular (and often most successful) characters played both before and after patch 5.13.

## 2 Features

### 2.1 What Got Done

- Graphical and charted display of change in win/pick rate per champion
- Charted display of change in champion roles
- Sortable by data type, game mode, region, & rate type

### 2.2 Our Ideas / Selected for Development

- Purchase/win rates of items with respect to champions
- Time of purchases (earlier/later 'power spikes')
- Look for any emerging meta picks
- More general game trends of before & after: first dragon, first baron, first tower, average kills at a certain time, etc.

### 2.3 Complications

- High computing time - there is A LOT of data to download and parse through
- Poor script testing - overnight scripts failed multiple times
- No time to work on this - both developers have other work to do

## 3 Backend

Although our web app is a static site, we used Django help us manage the database as we are both experience in Python and Django.

## 3.1 Database

We used SQlite for our databse. We were originally going to use PSQL, but neither of us could host and we did not want to spend money renting a server. The database schema was set up using Django models, and there are tables for match data, champions, and items.

The match table has columns for match_id, region, version, gamemode, and data. The data field is a json string dump of the data we receive from the API.

Both the champion and item tables have region, version, gamemode, picks, wins, and name.

## 3.2 Downloading

We ran Python scripts to download and populate the database with data required.

## 3.3 Counting

To count how mand picks/wins a champion or item has, we would iterate through each match of the specified region, version, and gamemode to count and save the picks and wins.

## 3.4 K-Means Clustering

A simple machine learning algorithm was implemented to see if a champion's in-game role differed between the two game versions.

## 3.5 Exporting

lorem ipsum

# 4 Frontend

## 4.1 Graphing

lorem ipsum

## 4.2 Charting

lorem ipsum