

**UMEC/Utah AHEC Primary Care Workforce Optimization -**

**Python Function Manual**

# Optimization functions (Python functions) user manual

## Overview

Given population needs and currently available provider mix, find optimal provider composition (i.e. types and number of providers). It uses three different optimization algorithms depending on goal, and each of them returns time allocation (i.e. in each service, which provider types spend how many minutes), and some additional statistics.

## Ideal provider staffing [option = ideal\_staffing]

## Purpose: Given population need, it finds an *optimal* provider team composition (i.e., types and number of providers)

## Use case: We have 100k population with 2k patients with diabetes and 1k patients with depression. How many (and what types of) providers do we need to have to provide best primary care?

Detail:

﻿It finds optimal time allocation based on wage and suitability score. Wage is mean/median wage of each provider type. ‘Suitability’ is placeholder for a variable that users want to consider to allocation time to each provider type besides wage. Some examples would be top/bottom of license or quality measure. ﻿Suitability can be any number between [0-1], and function converts the scale of suitability using ‘sut\_target’, which makes the most desirable score 0 and least desirable score 1. See sut\_target under input section for detail. The optimization is based on linear programing (python Pulp library).

Users can choose three sub-options.

all\_combination: it tries different weight combinations (0 to 1, with 0.1 interval) between wage and suitability. User provides wage weight, then suitability weight gets calculated (e.g. wage weight = 0.3 then suitability weight = 0.7). Outputs are total wage, total suitability, ideal provider composition, and detail F2F time allocation for each service x provider type. Note that ideal provider composition considers documentation and coordination time (overhead time). However, it won’t affect the ideal time allocation, thus for detail service x provider type time allocation, it only shows F2F time (FYI. thus stacked bar plot, such as “F2F Service allocation by provider type (Min)” or “F2F Service allocation by service type (Min)” only shows F2F time, but ”supply vs. needs “plot contains overhead time.)

wage\_max: users can set maximum available wage budget, and it returns ideal provider composition, total wage (below the wage\_max), total suitability score, and detail F2F time allocation for each service x provider type. If given wage\_max is smaller than total wage to support the population need, it provides a message with min/max total wage to serve the population needs, and users need to run the function again using wage between min/max total wage. Internally, first it runs “all\_combination” to find proper wage and suitability weight, then narrow the search. Thus, it takes longer computational time.

wage\_weight: users can provide wage\_weight (any number between 0 and 1). The weight for suitability then naturally 1 – wage\_weight. It returns ideal team composition, total wage, total suitability, and detail F2F time allocation for each service x provider type

## Ideal provider staffing constrained by current providers’ composition [option = ideal\_staffing\_current]

## Given population need and current providers’ composition, it returns an *optimal* provider team composition

## Use case: We have 100k population with 2k patients with diabetes and 1k patients with depression. We already have 2 physicians. How many (and what type) *more* providers do we need to have?

Detail:

﻿It finds optimal time allocation based on weight between wage and suitability, and currently available provider constrains. It first subtracts overhead time from available supply time, then for each provider types F2F is more than population need. If one of provider type has higher supply than population need, and then it can’t find an optimal allocation. Otherwise, it uses the same three sup-options that ideal\_staffing uses (all\_combination, wage\_max, wage\_weight) and returns ideal team composition, total wage, total suitability, and detail F2F time allocation for each service x provider type.

The optimization algorithm is similar to ideal\_staffing (i.e. based on linear programing) and use current supply as additional constrains.

## Ideal service/task allocation [option = service\_allocation]

## Given provider’s current composition, what would be an *optimal* task/service allocation

## Use case: We have 100k population with 2k patients with diabetes and 1k patients with depression. We have 2 physicians, 1 RN, 2 MA, which is not sufficient enough to support current population need but can’t hire new providers. How can we allocate services across providers to minimize over-allocation to certain provider type?

Detail: ﻿

It finds optimal time allocation to minimize the variance between current supply and need. First it calculates overhead time. If the overhead time is higher than current supply, it does not consider overhead time. Otherwise, it subtracts overhead time from the current supply time (i.e. number of provider x FTE times) and finds time allocation to minimize the variance between supply and demand. It does not consider wage nor suitability, and this time allocation is purely based on F2F time (i.e. avoid over-allocation). (FYI: As a potential extension, if wage or suitability are important, you can find weighted variance).

It uses quadratic programs (python CVXOPT library). Algorithm constructs P, q, G, h, A and b matrix, and solve following problem.



FYI: There are many resources, but I put very simple tutorial found from <https://courses.csail.mit.edu/6.867/wiki/images/a/a7/Qp-cvxopt.pdf>.)

## How to call optimization functions

## Input:

It needs .csv file containing following information (FYI, if you upload excel file to the server you can get following .csv files)

geo\_area\_list: it contains all geographic area that user can choose. It also contains social determinant health index (1 to 5, 1 being the healthiest) for each geographic area. (FYI, insides scale = 5.0 gets hard-coded. Thus if you use more detail health index such as 1-10, then please change the inside code)

Population: it shows the number of populations in each age, sex and geographic area. To calculate future preventive/chronic/acute needs, it relies on population in this tab. (FYI, thus ‘year’ should be found in this population tab).

provider\_list: it contains all provider\_list that one can choose. (FYI: If you want to remove some provider types, please delete them here. It you want to add additional provider types, please add one here, along with provider\_supply, service\_characteristics, overhead\_work

provider\_supply: it contains number of provider and wage in each geographic area and in each provider type. It also has provider\_growth\_rate and provider\_wage\_trend, which used to calculate future supply

Encounter\_type: it contains encounter type per three encounter categories (preventive, acute, and chronic)

Encounter\_detail: it contains in each encounter category, encounter type, service category, and service detail. Also, it shows the rate per encounter per year (i.e. population/patients do not need the service for every encounter, then rate per encounter gets <1)

Service\_characteristics: each service category and service description, it contains minimum and maximum F2F time, along with suitability score. It calculates F2F time using minimum and maximum F2F time weighted by social determinant health index (from geo\_area\_list). Suitability score is assigned to providers who are eligible to deliver the service. If license does not allow a provider type delivers a service, then it should be NA (FYI, sometimes white blank can cause error – whole columns may be considered as string rather than numeric - thus across tab, please use NA or zero depending on context. For instance, overhead time, MA does not do documentation, then it should be zero. But if MA should not deliver a service, then it should be NA.)

Overhead\_work: it contains time to be spent for documentation or coordination. prop\_f2f\_tot indicates total time proportional to F2F time (e.g. Documentation prop\_f2f\_tot = 1 and phy = 0.25 means time for documentation = 1\*total F2F time, and physicians spend 0.25\* 1\*total F2F time.

Pop\_prev\_need: it contains preventive care services (column), and frequency of each age and sex.

Pop\_acute\_need: it contains acute care services (column), and prevalence (out of 1000) of each age and sex.

pop\_chronic\_prev: for each age and sex, it shows prevalence (out of 1000 people).

pop\_chronic\_trend: for each age and sex, chronic disease prevalence trend. When users want to understand future population demand, it will use this trend to update the chronic patient population. (e.g. anxiety trend = 0.01 then year = 2020 and current year = 2018, then updated anxiety population = (1 + 0.01)^2\*(anxiety population in 2018. All future supply/need using same logic, i.e. (1+growth rate)^(year difference) \* current demand/supply

Besides these .csv files, users also provide following inputs.

﻿geo: area one wants to analyze; one of area shown in the geo\_area\_list

year: year one wants to analyze (i.e. target year)

current\_year: current year. If current\_year = year, then it will use current year demand/supply. Otherwise, it will estimate demand and supply of the target year. For preventive care, it will get number of population from the target year, for chronic care, we will update based on pop\_chronic\_trend, and for acute care, we use the same number (assume that the prevalence won’t change much). For provider and supply and wage also changed (based on provider\_supply, provider\_growth\_trend and provider\_wage\_trend).

option: one of three main functions, i.e. 'ideal\_staffing', 'ideal\_staffing\_current', 'service\_allocation'

subpotion: for 'ideal\_staffing' and 'ideal\_staffing\_current', users can further choose "all\_combination", "wage\_max", or "wage\_weight". For service\_allocation, sub\_option = None (or function manually change sub-option to None)

sub\_option\_value: for sub\_option, wage\_max = maximum wage, wage\_weight = wage weight. Note that wage\_max should be a number (with out $) and weight should be number (float) between 0.0 and 1.0

sut\_target: Ideal suitability target. Currently suitability indicates top & bottom of the license - easy task/bottom of license (0.0) and difficult task/top of license (1.0) - and want to avoid top/bottom of the license, and we set default target 0.8. The function converts suitability score in a way that 0.8 has the smallest number (since we want to minimize the suitability score) If you think specific population segments need more skilled provider, then you can use higher sut\_target. If it is quality score and 1 is the 'best' quality, then target should be 1; and 0 is the best score then the target should be 0.

collapse\_group: this can be used for 'service\_allocation'. If same types of providers can provide services belongs to the same encounter\_category and srv\_category, the it collapses all services and allocate time. Then the time redistributed to each service proportionally. It helps computational speed. Current default is False. FYI, if you have more services, and takes long computational time, you can use this option.

FTE\_time: 60x2080, used to convert FTE to min per year

## Output

For options 'ideal\_staffing' or 'ideal\_staffing\_current', they return following information

total\_wage: sum of wage of all providers

total\_sutab: sum of wage of all providers

FTF: sum of population need for each provider (F2F + overhead)

detail\_f2f\_mini: F2F time allocation (minutes) for each provider type x each service

FYI, Under wage\_weight, or wage\_max I also calculated

ind\_wage: sum of wage for each provider type. Some of ind\_wage = total\_wage

I thought that this is not useful info since you know mean wage

ind\_sutab: sum of sutability for each provider type. Some of ind\_sutab = total\_sutab. This one, I was not sure ‘total sum’ makes sense. Perhaps mean sutability be better depending on how you define sutability

'service\_allocation'

FTF: Current needs to minimize supply vs. needs gap. Needs can be F2F or F2F + overhead (i.e. if current supply can’t support all overhead time, then it minimize F2F only, if it can then minimize needs vs. supply time – overhead.

detail\_f2f\_min: F2F time allocation (minutes) for each provider type x each service

Few plots and summary stats.