Derrick Priebe

Matthew Peters

Pierre Augustamar

INFX 575

Final Paper

**An Analysis of the linkages between Exercise and Property in King County**

**Questions:**

*State the question you sought to answer, as clearly as possible.*

We analyze the property and exercise data in two distinct ways:

1. **Do exercise routes affect property values?**

Why: To understand whether there is a strong case with regards to property value for community development of more formal exercise routes.

**2) What is behind what routes people choose for exercise?**

Why: From a runner’s perspective, try to identify main characteristics that can be identified that would accurately align with chosen routes.

**Related Work:**

*What other approaches existed for this problem, if any? How are you different?*

There are some previous work with regard to this topic as detailed below:

The **paper “Not Just a Walk in the Park: Methodological Improvements for Determining Environmental Justice Implications of Park Access in New York City of the Promotion of Physical Activity”** takes a focused look at a single city to analyze availability of park to various socioeconomic and ethnic sub-groups. While we have academic interests that are similar, the scope and approach of our study is quite different. This may be a relevant read simply to provide academic and methodological insights, although it is not likely to have anything in common with means of research. As the paper notes, “This study is designed to shed light on the“unpatterned inequities” of park distributions identified in previous studies of New York City park access. (Miyake et al)”.

Source: Miyake KK, Maroko AR, Grady KL, Maantay JA, Arno PS. Not Just a Walk in the Park: Methodological Improvements for Determining Environmental Justice Implications of Park Access in New York City for the Promotion of Physical Activity. *Cities and the environment*. 2010;3(1):1-17.

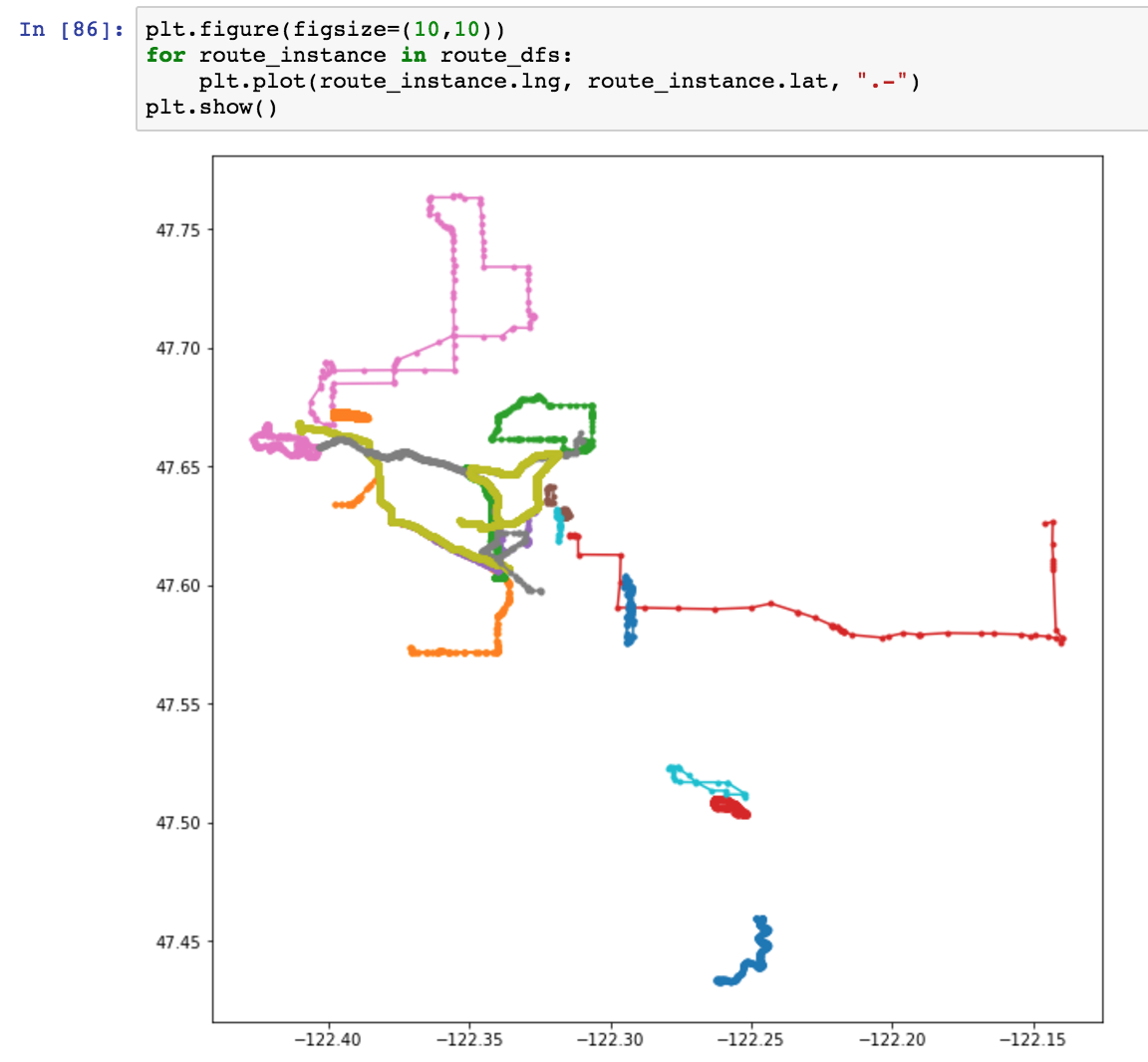
A **paper, “City, Culture and Society”, by Lawson and Fadare** uses “simple random sampling of household heads” (Lawson, Fadare) in three distinct neighborhoods of Eti Osa, Nigeria. Here, our methodologies and score vary greatly. As noted in the review: “This paper considers the effects of socio-economic status as a determinant of urban health outcomes. Issues examined include housing and environmental conditions as well as socio-economic characteristics such as age, gender, income and household size. Furthermore, health seeking behaviour was investigated and these include expenditure on health as well as health and nutritional habits.”  
Source: Taibat Lawanson, Samson Fadare, 2014, Elsevier Ltd, City, Culture and Society, Volume 6, Issue 1, Pages 43-52, *Environment and health disparities in urban communities: Focus on Eti Osa, Nigeria*

**Data:**

*Describe the data you used to answer the question.*

We utilized wearable activity data as one main data source. Principally, we acquired exercise data from MapMyRun API via JSON in May 2017 over 2 week period. This provides workout information including location, run length, run speed, and other applicable running information. We successfully extracted this data with sample as shown in figure 1 below. It is inclusive of King County routes only. Route creation covers years 2005 - 2017. The exercise dataset is not a complete dataset because there were limitations on amount of data per day, there were technical data flow outages, and we ran out of time in the quarter overall for the data collection process.

***Figure 1: Sample MapMyRun Routes***



Sample MapMyRun route data in JSON form:

{

"description": "",

"country": "us",

"total\_descent": -4.0657231389,

"images": [],

"state": "WA",

"\_links": {

"alternate": [

{ "href": "/v7.1/route/100001/?format=kml&field\_set=detailed",

"id": "100001",

"name": "kml" },

{ "href": "/v7.1/route/100001/?format=gpx&field\_set=detailed",

"id": "100001",

"name": "gpx" }

],

"self": [ { "href": "/v7.1/route/100001/", "id": "100001" } ],

"privacy": [ { "href": "/v7.1/privacy\_option/3/", "id": "3" } ],

"activity\_types": [ { "href": "/v7.1/activity\_type/16/", "id": "16" } ],

"thumbnail": [{ "href": "//drzetlglcbfx.cloudfront.net/routes/thumbnail/100001/1155832438?size=100x100" } ],

"user": [ { "href": "/v7.1/user/18717/", "id": "18717" } ]

},

"city": "Seattle",

"min\_elevation": 3.32,

"postal\_code": "98188",

"points": [

{ "dis": 0.0,

"ele": 5.83,

"lng": -122.247126102,

"lat": 47.4576162962 },

{ "dis": 29.12,

"ele": 6.61,

"lng": -122.24719584,

"lat": 47.4573587734},

...

{"dis": 9987.45,

"ele": 5.24,

"lng": -122.247072458,

"lat": 47.4578339202},

{"dis": 10009.96,

"ele": 5.82,

"lng": -122.24714756,

"lat": 47.4576380587}],

"total\_ascent": 7.0583606896,

"data\_source": "MapMyRun",

"created\_datetime": "2006-08-17T16:33:58+00:00",

"start\_point\_type": "",

"name": "The Wilbur 10K",

"max\_elevation": 10.62,

"distance": 10010.0,

"updated\_datetime": "2006-08-17T16:33:58+00:00",

"climbs": [],

"starting\_location": { "type": "Point",

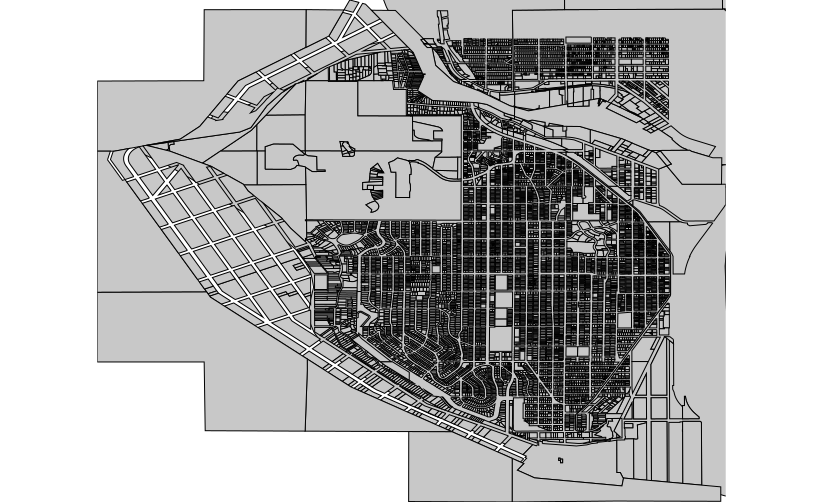
"coordinates": [ -122.2471261024, 47.4576162962 ]

}

}

As our secondary source we utilized is real estate information. We utilized data from King County GIS records. We successfully extracted this data from their FTP store at <ftp://ftp.kingcounty.gov/gis-web/GISData/property_SHP.zip>. An example of property information is visualized below in figure 2. The property data is entirely inclusive of King County only. Properties exist in 40 cities across the county.

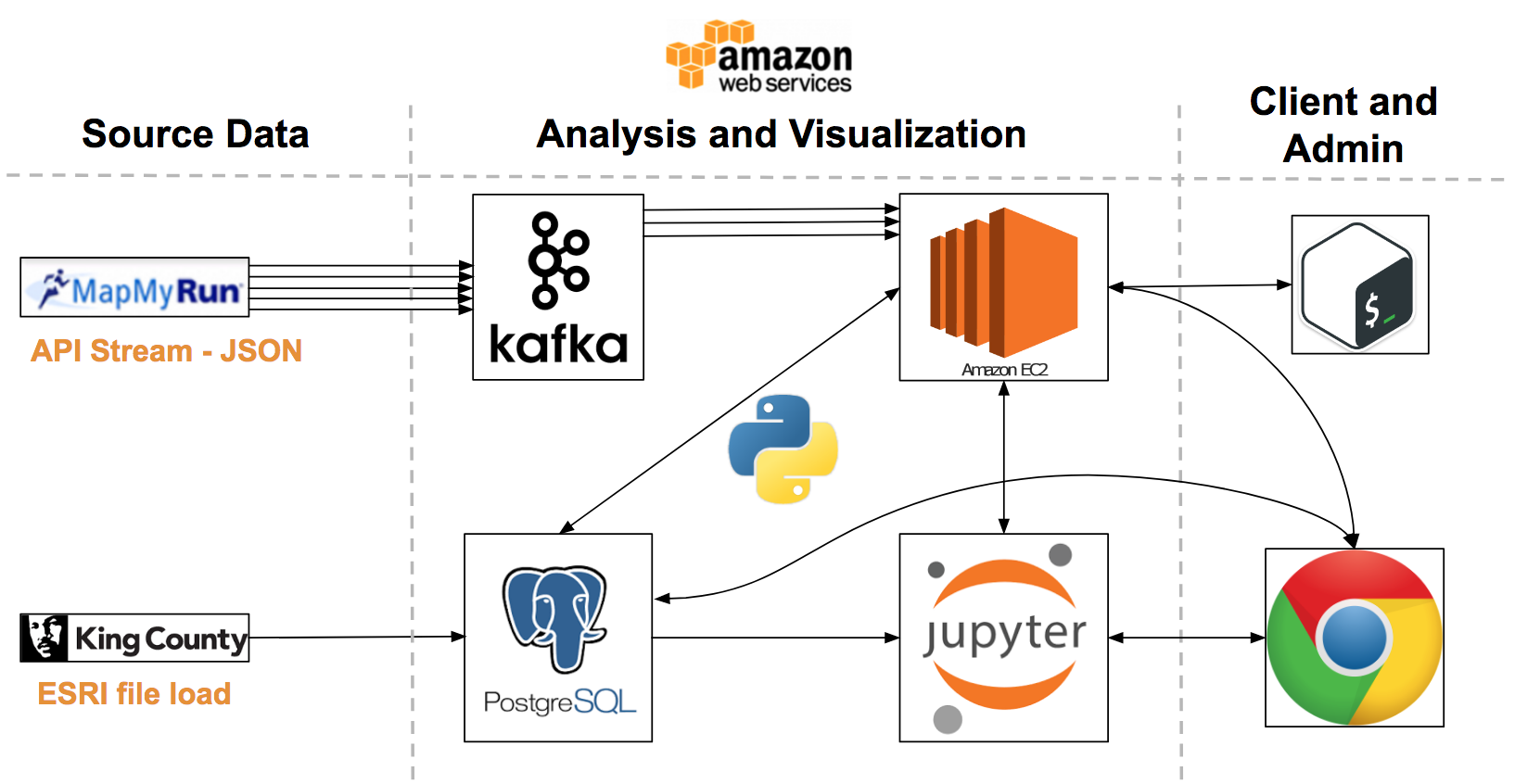
***Figure 2: Magnolia Property Map***



**Technology:**

**Overall Architecture**

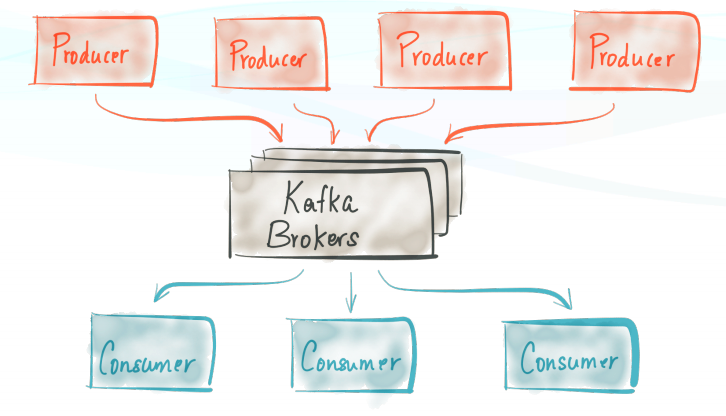
We utilized AWS as the the main data store. Within AWS, we will utilize Apache Kafka as a main ***pipeline*** to secure regular API data extracts from MapMyRun which feeds PostgreSQL database implementing PostGIS extensions and running on AWS RDS. Overall, we utilized PostgreSQL (***schema*** *detail below*) as the main source to hold tables from all the utilized data sources. Within AWS, Python, utilizing Jupyter Notebook, is our data analysis tool of choice where we will pickled and pulled applicable data frames, make necessary transformations, and visualized the data.

***Figure 3: Architecture***

**Pipeline Detail**

For data persistence, we utilized Apache Kafka which is an open source distributed pub/sub messaging system (see figure 4).

***Figure 4: Apache Kafka architecture***



We built our kafka hub with two topics. One topic contains the mapmyrun data which is taken in real-time. And a second topic contains static home pricing values. Thus, we have two producers pushing data to the Kafka brokers. The data stored in Kafka then goes to a transformation that is handled by the consumers. The transformed data will be uploaded into a storage database. At this time, we only anticipate having one consumer, but as we investigate additional open source tools, we may add other consumers if time permits and or if it adds values to the project.

To set-up a communication layer between Kafka and the related producers, and consumers, we used PyKafka. PyKafka is a cluster-aware Kafka client for Python. It includes Python implementations of Kafka producers and consumers.

To monitor the data in Kafka, we used Presto which is an open source distributed SQL query engine for running interactive analytic queries against data sources. A topic in Kafka is considered to be a table in Presto. Thus, Presto allows to run SQL like query against a topic.

**Database Schema Detail**

"**public.city\_county**"

Column | Type | Modifiers

----------------+--------+-------------------------

city | text |

county | text |

city\_county\_id | bigint | not null default nextval('city\_county\_city\_county\_id\_seq'::regclass)

Indexes:

"city\_county\_pkey" PRIMARY KEY, btree (city\_county\_id)

Referenced by:

TABLE "neighborhoods" CONSTRAINT "neighborhoods\_city\_county\_id\_fkey" FOREIGN KEY (city\_county\_id) REFERENCES city\_county(city\_county\_id)

"**public.route**"

Column | Type | Modifiers

-----------------+----------------------------+------------------

route\_id | bigint | not null default nextval('route\_route\_id\_seq'::regclass)

map\_my\_run\_id | character varying |

postal\_code | character varying |

city | character varying |

neighborhood\_id | bigint |

distance | bigint |

create\_datetime | timestamp with time zone |

route\_name | character varying |

total\_ascent | character varying |

max\_elevation | double precision |

thumbnail | character varying |

path | geometry(LineStringZ,4362) |

city\_county\_id | bigint |

Indexes:

"route\_pkey" PRIMARY KEY, btree (route\_id)

"route\_path\_idx" gist (path)

Foreign-key constraints:

"route\_city\_county\_id\_fkey" FOREIGN KEY (city\_county\_id) REFERENCES city\_county(city\_county\_id)

"**public.parcel\_address**"

Column | Type | Modifiers

----------------+------------------------+---------------------

gid | integer | not null default nextval('parcel\_address\_gid\_seq'::regclass)

major | character varying(6) |

minor | character varying(4) |

pin | character varying(10) |

comments | character varying(254) |

sitetype | character varying(2) |

alias1 | character varying(60) |

alias2 | character varying(60) |

siteid | numeric |

addr\_hn | character varying(10) |

addr\_pd | character varying(2) |

addr\_pt | character varying(6) |

addr\_sn | character varying(80) |

addr\_st | character varying(6) |

addr\_sd | character varying(2) |

addr\_num | numeric(10,0) |

addr\_full | character varying(120) |

fullname | character varying(120) |

zip5 | character varying(5) |

plus4 | character varying(4) |

ctyname | character varying(28) |

postalctyn | character varying(28) |

lat | numeric |

lon | numeric |

point\_x | numeric |

point\_y | numeric |

county | character varying(12) |

kroll | character varying(4) |

juris | character varying(2) |

big\_ten | character varying(20) |

budget\_uni | character varying(40) |

kctp\_city | character varying(75) |

kctp\_state | character varying(2) |

plss | character varying(11) |

prop\_name | character varying(50) |

plat\_name | character varying(50) |

plat\_lot | character varying(14) |

plat\_block | character varying(7) |

presentuse | integer |

lotsqft | numeric(10,0) |

levycode | character varying(4) |

levy\_juris | character varying(25) |

new\_constr | character varying(1) |

taxval\_rsn | character varying(2) |

apprlndval | numeric |

appr\_impr | numeric |

tax\_lndval | numeric |

tax\_impr | numeric |

accnt\_num | character varying(12) |

kctp\_taxyr | integer |

kctp\_par | character varying(10) |

unit\_num | character varying(254) |

bldg\_num | character varying(254) |

condositus | character varying(254) |

qts | character varying(2) |

sec | character varying(2) |

twp | character varying(2) |

rng | character varying(2) |

primary\_ad | integer |

legaldesc | character varying(254) |

shape\_area | numeric |

shape\_len | numeric |

geom | geometry(MultiPolygon) |

city\_county\_id | bigint |

Indexes:

"parcel\_address\_pkey" PRIMARY KEY, btree (gid)

"parcel\_address\_city\_idx" btree (ctyname)

"parcel\_address\_geom\_idx" gist (geom)

Foreign-key constraints:

"parcel\_address\_city\_county\_id\_fkey" FOREIGN KEY (city\_county\_id) REFERENCES city\_county(city\_county\_id)

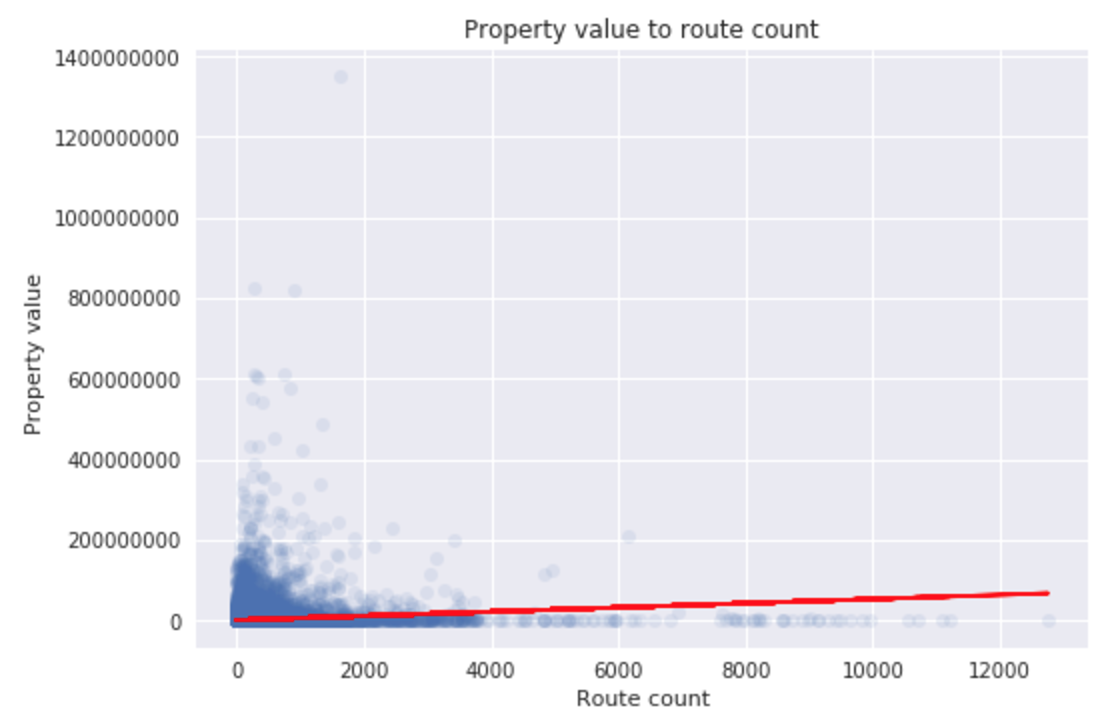
**Methods and Approach:**

*How did you obtain your answer? What challenges did you face?*

In order to obtain our answer we made some transformations and performed analysis. For transformations, we consolidated property use classes from a list of over 100 to a consolidated list of 6 in total. We used this as one of our independent variables. We also created a variable called “lot sq ft vs city sq ft” which was meant to incorporate the size of a property in comparison to the total city. Other than some transformations, we ran a standard linear regression for each of our two scenarios - property value as dependent variable and route count as dependent variable.

For focus on **exercise routes and property values**, fist a correlation plot was utilized to compare at a general level the relationship between the two variables of property value and route count/presence:

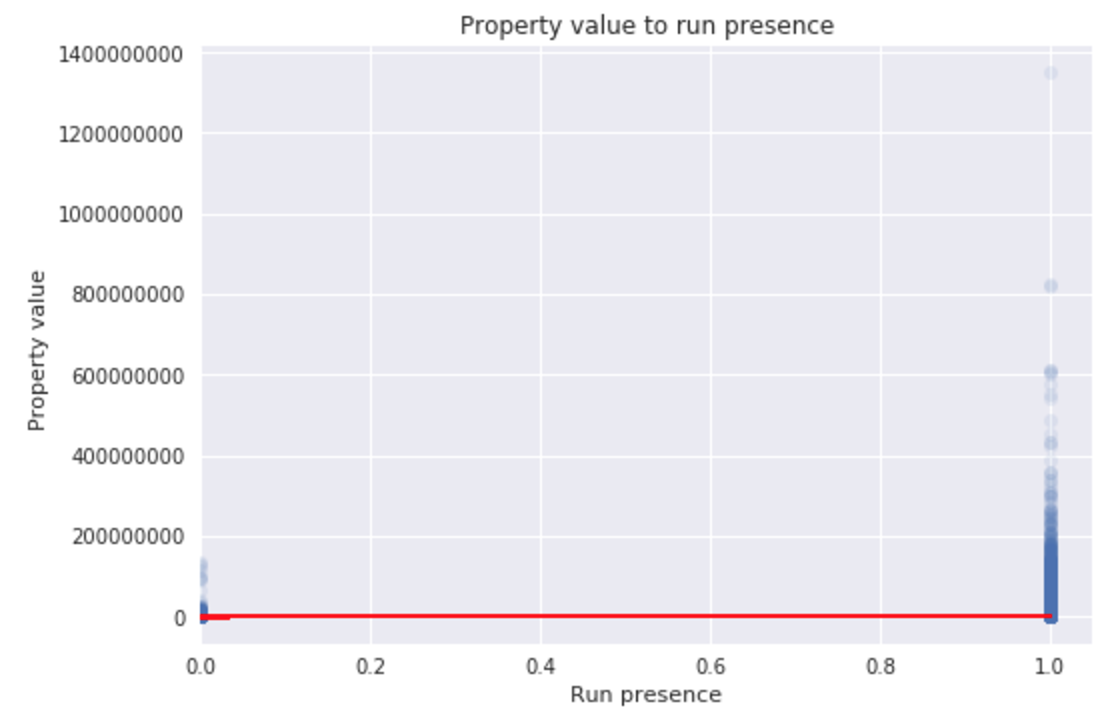
**Total property value vs Route count:**



Slope = 585,275; Rsquared = .04; Pvalue = <.05

The slope of 585,275 does show a positive relationship. The p-value confirms that you can reject the null hypothesis. However, the r squared value of 4% is low and means that model only explains 4% of the variability of the response data around its mean.

**Total property value vs Route presence:**

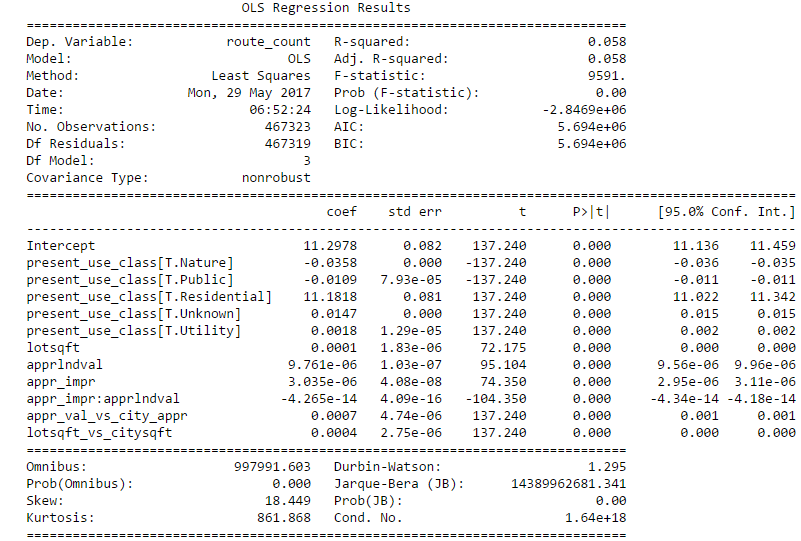


Slope = 5,234; Rsquared = .01 ; Pvalue = <.05

The slope of 5,234 does show a positive relationship. The p-value confirms that you can reject the null hypothesis. However, the r squared value of 1% is low and means that the model only explains 1% of the variability of the response data around its mean.

For the regression analysis, we used total property value as the dependent variable, and Use class, lot sq ft, route count, route presence, and lot sq ft city sq ft ratio as the independent variables. Looking at the coefficients, we found that route presence, route count, lotsqft had the highest positive correlations; commercial and public properties also had the highest correlations of use class. Again the p-value indicated we could reject the null hypothesis. However, while better, our r squared value showed that the that model only explains 23% of the variability of the response data around its mean.

For focus on **exercise routes and property variables overall**, we incorporated a standard regression analysis. The dependent variable was route count and the independent variables were Use class, lot sq ft, route count, land value, improvement value, improvement to land value ratio, and lot sq ft city sq ft ratio. See below OLS result.



Each of the controls in this model will have an impact on the quantity of running routes. As we add or remove control variables the point estimate of the running routes will be affected. The intercept, in this linear regression, is essentially the expected value required to experience any form of impact on the running routes when considering multiple variables such as the type of property is located next to the running route, the home values, and the average lot square feet of the properties. In this case, it takes on average 11.2978 for running routes in an area to see any significant changes. The regression summary shows shows the intercept at 11.2978 and the beta arrays at -0.0358, -0.0109, 11.1818, 0.0147, 0.0018, 0.0001, 9.76e-06, 3.05e-06, -4.265e-14, 0.0007and 0.0004 respectively.

standard error: This gives an estimate of the expected difference in case we run the model again and again. In other words, we can say that the residential properties in order to affect the quantity of running routes vary by a std error of 0.081. Also, the standard error can be used to compute confidence intervals and to statistically test the hypothesis of the existence of a relationship between enrollment (increase or decrease) and a given treatment.

R-squared: This statistic measures how successful the fit is in explaining the variation of the data. R-square can take on any value between 0 and 1, with a value closer to 1 indicating that a greater proportion of variance is accounted for by the model. In this model, we got an R-square value of 0.058 means that the fit explains 5.8% of the total variation in the data.

Adj. R-squared: This statistic uses the R-square statistic defined above, and adjusts it based on the residual degrees of freedom. The residual degrees of freedom is defined as the number of response values n minus the number of fitted coefficients m estimated from the response values. In this model, we have adj R-squared that's 0.041 which indicates that this model is far from being a better fit. Thus, further analysis would need to be done in order to figure out what part of the data set is affecting our model.

Coef: These are the values for the logistic regression equation for predicting the dependent variable from the independent variable. A generics prediction equation for this model will be as follows:

Y = 11.2978 - 0.0358\*Nature - 0.0109\*Public + 11.1818\*Residential + 0.0147\*Unknown + 0.0018\*Utility + 0.0001\*lotsqft + 9.76e-06\*apprlandval + 3.05e-06\*appr\_impr - -4.265e-14\*appr\_impr :apprlndval + 0.0007\*appr\_val\_vs\_city\_appr + 0.0004\*lotsqft\_vs\_citysqft

t-statistic: We use the t-test to test the null hypothesis that the coefficient of a given predictor variable is zero, implying that a given predictor has no appreciable effect on the response variable. The alternative hypothesis is that the predictor does contribute to the response. Our model generated a t-statistic value of 137.240 for the slope.

P>|t|: This column shows the 2-tailed p-values used in testing the null hypothesis that the coefficient (parameter) is 0. For this model, we are using an alpha of 0.05. For our model, the coefficient for all of the beta arrays are statistically significant because their p-value of 0.000 is less than the threshold set at 0.05.

**T-Test** **analysis**

The null hypothesis for this model is that there is no difference in the home values between properties that were built around available routes and properties that were built without routes. To answer this question, we calculated the t-test difference between properties with routes and properties without routes.



The t-test generated a p-value that's vastly less than 0.05, thus we reject the null hypothesis that there is no difference between properties built around running routes and those built without running routes around.

**Challenges:**

We faced several challenges along the way. First, setting up a reliable data flow took some time. Initially our Kafka instances failed several times which required us to revise the code and approach several times to ensure a constant flow of unique records. On the database side, geolocation transformations proved to be complex. The amount of data that we needed to utilize and process required required cloud processing to ensure adequate power. The amount of data also gave us issues with standard data frame creation techniques. We had to investigate and use the pandas pickling process to package and read data efficiently.

**Results:**

*What was the answer to your question? If the results were inconclusive, say why.*

Our two hypotheses attempted to examine if exercise routes affect property values whether property related variables affect the routes people choose for exercise. For both, utilizing p-values, we are able to reject the null hypothesis. There is some evidence of positive correlation between select variables for both analyses. However, the r-squared value was low enough that model only explains a small percentage of the variability of the response data around its mean. As a result, the results are inspiring, but inconclusive as constructed. Further analyses with additional independent variables incorporated is required to more thoroughly inspect the topic.

**Limitations:**

*What caveats and limitations are there on your answer? What potential biases may influence your results? How generalizable are all your results? Could there be any possible misuse of your results that you should guard against?*

Our results come with some caveats and limitations. First, our dataset related to exercise is not complete so it should be completed prior to further analysis. Many biases could have influenced our results. For example, the preponderance of trails or public infrastructure in certain areas could have had an effect on routes and property values. Another example, which we tried to control for in our follow up exercise, is the impact of individual neighborhoods or zip codes themselves on the outcome. Our results are fairly specific to the problem at hand, but could be generalized somewhat to address any property or exercise related information. Given the caveats provided and recommendation for further analysis, I don’t believe that there could be any reasonable expected misuse.

**Future work:**

*What extensions would you make if you were given the opportunity?*

There are a few things we would like to do in the future to incorporate a more complete and conclusive product. First, we would need to access more complete exercise dataset to have an all inclusive set of information to work with. We also need to integrate more independent variables such as distance to city center cores, distance to schools, distance to water, and others that could help find more meaningful correlations. Both these would take considerable time and effort beyond that afforded to us during the academic quarter.

Some potential uses of our model could include: advertisers targeting ads to home buyers who are also fitness enthusiasts; affect how municipalities design building code; policy based on a set of machine learning that utilizes this model; or add into Zillow’s Zestimate algorithm.

**Appendix:**

**Code snippets**

See bolded section for a short note about the action of the specific code.

|  |
| --- |
| """  **route\_writer.py**  @author: Matthew Peters  @organization: University of Washington INFX-575 Spring 2017  @license: (c) 2017, Matthew Peters  **Polls kafka at the environmentally configured hosts, and topics, transforms MapMyRun route JSON objects**  **to database scheme and inserts route records to environmentally configured AWS PostgreSQL hosted on RDS.**  """  import pykafka  import json  import psycopg2  from os import environ, path  from datetime import datetime  """  MAP MY RUN ROUTE FORMAT  {  "description": "",  "country": "us",  "total\_descent": -4.0657231389,  "images": [],  "state": "WA",  "\_links": {  "alternate": [  { "href": "/v7.1/route/100001/?format=kml&field\_set=detailed", "id": "100001", "name": "kml" },  { "href": "/v7.1/route/100001/?format=gpx&field\_set=detailed", "id": "100001", "name": "gpx" } ],  "self": [ { "href": "/v7.1/route/100001/", "id": "100001" } ],  "privacy": [ { "href": "/v7.1/privacy\_option/3/", "id": "3" } ],  "activity\_types": [ { "href": "/v7.1/activity\_type/16/", "id": "16" } ],  "thumbnail": [ { "href": "//drzetlglcbfx.cloudfront.net/routes/thumbnail/100001/1155832438?size=100x100" } ],  "user": [ { "href": "/v7.1/user/18717/", "id": "18717" } ]  },  "city": "Seattle",  "min\_elevation": 3.32,  "postal\_code": "98188",  "points": [  {  "dis": 0.0,  "ele": 5.83,  "lng": -122.247126102,  "lat": 47.4576162962  },  ],  "total\_ascent": 7.0583606896,  "data\_source": "MapMyRun",  "created\_datetime": "2006-08-17T16:33:58+00:00",  "start\_point\_type": "",  "name": "The Wilbur 10K",  "max\_elevation": 10.62,  "distance": 10010.0,  "updated\_datetime": "2006-08-17T16:33:58+00:00",  "climbs": [],  "starting\_location": { "type": "Point", "coordinates": [ -122.2471261024, 47.4576162962 ] }  }  POSTGIS LINESTRING FORMAT  LINESTRING(1 2 3,3 4 5,6 6 6)  """  def route\_to\_linestring(route):  """  Extract positional data from a route and convert it to a postgis linestring  :param route: dict  :return: str  """  points\_list = route.get("points")  points = ["{} {} {}".format(point["lng"], point["lat"], point.get("ele", 0)) for point in points\_list]  return "LINESTRING(" + ",".join(points) + ")"  """  "​ public.route​ "  Column | Type | Modifiers  -----------------+----------------------------+------------------  route\_id | bigint | not null default nextval('route\_route\_id\_seq'::regclass)  map\_my\_run\_id | character varying |  postal\_code | character varying |  city | character varying |  neighborhood\_id | bigint |  distance | bigint |  create\_datetime | timestamp with time zone |  route\_name | character varying |  total\_ascent | character varying |  max\_elevation | double precision |  thumbnail | character varying |  path | geometry(LineStringZ,4362) |  city\_county\_id | bigint |  Indexes:  "route\_pkey" PRIMARY KEY, btree (route\_id)  "route\_path\_idx" gist (path)  Foreign-key constraints:  "route\_city\_county\_id\_fkey" FOREIGN KEY (city\_county\_id)  REFERENCES city\_county(city\_county\_id)  """  def convert\_route(json\_route):  """  Convert the JSON version of a route to a set of parameters appropriate for inserting into  the PostgreSQL database.  :param json\_route: dict:  :return: params: list:  """  map\_my\_run\_id = json\_route.get("\_links",{"self":{"id": None}}).get("self")[0].get("id")  postal\_code = json\_route.get("postal\_code")  city = json\_route.get("city").strip()  neighborhood\_id = None  distance = json\_route.get("distance")  create\_datetime = json\_route.get("created\_datetime")  route\_name = json\_route.get("name", map\_my\_run\_id)  total\_ascent = json\_route.get("total\_ascent")  max\_elevation = json\_route.get("max\_elevation")  thumbnail = json\_route.get("\_links", {"thumbnail":[{"href": None}]}).get("thumbnail")[0].get("href")  path = route\_to\_linestring(json\_route)  return [str(map\_my\_run\_id),  postal\_code,  city.upper(),  neighborhood\_id,  distance,  create\_datetime,  route\_name,  total\_ascent,  max\_elevation,  thumbnail,  path,  ]  def db\_insert(params):  """  Insert the parameters to the route table  :param params:  :return:  """  log\_file = path.join(environ.get("RW\_WORKING\_DIR"), environ.get("RW\_DB\_LOG"))  with open(log\_file, "a") as db\_log\_fh:  dsn = {"database": environ.get("RDS\_DB", "infx575"),  "host": environ.get("RDS\_HOST"),  "user": environ.get("RDS\_USER", "data\_service"),  "password": environ.get("RDS\_PASSWORD")  }  conn = psycopg2.connect(\*\*dsn)  with conn.cursor() as curs:  sql = curs.mogrify(  ("INSERT INTO route( map\_my\_run\_id, postal\_code, city, neighborhood\_id, distance, create\_datetime, "  "route\_name, total\_ascent, max\_elevation, thumbnail, path) "  "VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, st\_setsrid(st\_geomfromewkt(%s), 4362))"),  params)  try:  curs.execute(sql)  conn.commit()  except Exception as exn:  conn.rollback()  db\_log\_fh.write("{} ERR: {}\n{}\n{}\n".format(datetime.strftime(datetime.now(), "%Y-%m-%d %T"),  exn,  params,  sql))  finally:  conn.close()  def handle\_event(message):  """  Take the message and do with it what must be done  :param message:  :return:  """  value = None  event\_log = path.join(environ.get("RW\_WORKING\_DIR"), environ.get("RW\_EVENT\_HANDLER\_LOG"))  with open(event\_log, "a") as event\_log\_fh:  try:  value = str(message.value, 'utf-8')  j\_route = json.loads(value)  params = convert\_route(json\_route=j\_route)  except Exception as exn:  event\_log\_fh.write("{} ERR: {} {}\n".format(datetime.strftime(datetime.now(), "%Y-%m-%d %T"),  value,  exn)  )  return  try:  db\_insert(params)  event\_log\_fh.write("{} EVENT: {}\n".format(datetime.strftime(datetime.now(), "%Y-%m-%d %T"),  params[0])  )  except Exception as exn:  event\_log\_fh.write("{} ERR: {} {}\n".format(datetime.strftime(datetime.now(), "%Y-%m-%d %T"),  params[0],  exn)  )  def listen\_for\_messages():  """  :return:  """  consumer = None  host\_name = environ.get("KAFKA\_HOSTS")  log\_file = path.join(environ.get("RW\_WORKING\_DIR"), environ.get("RW\_EVENT\_LOG"))  with open(log\_file, "a") as event\_log\_fh:  event\_log\_fh.write("START TIME: {}".format(datetime.strftime(datetime.now(), "%Y-%m-%d %T")))  while True:  event\_log\_fh.write("CONNECTING TO {}\n".format(host\_name))  k\_client = pykafka.KafkaClient(hosts=host\_name, use\_greenlets=True)  topic\_name = environ.get("KAFKA\_TOPIC").encode()  event\_log\_fh.write(" TOPIC: {}\n".format(topic\_name))  try:  topic = k\_client.topics[topic\_name]  consumer = topic.get\_simple\_consumer()  except Exception as exn:  print("ERR -- SEE LOGS")  event\_log\_fh.write(  "{} ERR GETTING TOPIC: {} \n".format(  datetime.strftime(datetime.now(), "%Y-%m-%d %T"),  exn))  try:  for message in consumer:  handle\_event(message)  except Exception as exn:  print("ERR -- SEE LOGS")  event\_log\_fh.write("{} ERR: {} \n".format(datetime.strftime(datetime.now(), "%Y-%m-%d %T"),  exn)  )  if \_\_name\_\_ == "\_\_main\_\_":  listen\_for\_messages() |
|  |
|  |

|  |
| --- |
| """  **producer.py**  @author: Pierre Augustamar  @organization: University of Washington INFX-575 Spring 2017  @license: (c) 2017, Pierre Augustamar  **Sending data to kafka broker**  """  from pykafka import KafkaClient  from os import environ  from .mapmyrun import MapMyRun  import json  class Producer:  """Class for sending data to a Kafka broker.  :param host: The kafka host.  :param port: The kafka port.  """  def \_\_init\_\_(self, host, port, topic):  """    :param host:  :param port:  :params topic:  """  self.host = host  self.port = port  self.client = KafkaClient("{}:{}".format(self.host, self.port))  self.topic = self.client.topics[topic]  def sync\_produce(self, payload):  """    :param topic:  :param payload:  :return:  """  with self.topic.get\_sync\_producer() as producer:  producer.produce(payload)  def async\_produce(self, payload):  """    :param topic:  :param payload:  :return:  """  with self.topic.get\_producer(min\_queued\_messages=1, delivery\_reports=True) as producer:  producer.produce(payload)  while True:  try:  msg, exc = producer.get\_delivery\_report(block=False)  if exc is not None:  print('Failed to deliver msg {}: {}'.format(  msg.partition\_key, repr(exc)))  else:  print('Successfully delivered msg {}'.format(msg.partition\_key))  except Exception as exn:  print("ERR: {}".format(exn))  break  if \_\_name\_\_ == '\_\_main\_\_':  """  Script expects the following environmental variables:    KAFKA\_TOPIC  PRODUCER\_CITIES  KAFKA\_HOST  KAFKA\_POST  """  topic\_name = environ.get("KAFKA\_TOPIC")  cities = environ.get("PRODUCER\_CITIES")  producer = Producer(host=environ.get("KAFKA\_HOST") , port=environ.get("KAFKA\_PORT"), topic=topic\_name)  with open(cities, "r") as cities:  for city in cities:  mmr = MapMyRun(city)  for payload in mmr:  message = json.dumps(payload, indent=2)  producer.async\_produce(message) |

|  |
| --- |
| """  **mapmyrun.py**  @author: Pierre Augustamar  @organization: University of Washington INFX-575 Spring 2017  @license: (c) 2017, Pierre Augustamar  **Making a call to mapmyrun api to retrieve routes data**  """  import json  import requests  import pandas as pd  import sys  def get\_routesData(access\_token, key, city):  min\_dist = 1  public\_routes\_url = "https://oauth2-api.mapmyapi.com/v7.1/route/"  headers = {"Api-Key": key,  "Authorization": "Bearer {}".format(access\_token),  "Content-Type": "application/json"}  data = [{  "minimum\_distance": min\_dist},  {"city": city,  "state": "WA",  "country": "US",  "field\_set": "detailed"}  ]  resp = requests.get(public\_routes\_url, params=data[1], headers=headers)  if resp.status\_code == 200:  data = resp.json()  else:  print(resp.content) #throw error  #return json.dumps(data, indent=2)  return data  def get\_tokenAndKey():  # Application Credentials - these are secret!  Application = "INFX574A"  Key = "y6xqrts5cjnfydbaq7mhmd2eazu2vd4x"  Secret = ""  # We need a session cookie - so log in and request one  token\_endpoint = "https://api.ua.com/v7.1/oauth2/access\_token/"  grant\_type = "client\_credentials"  content\_type = "application/x-www-form-urlencoded"  action = "POST"  req = requests.post(url=token\_endpoint,  data={"grant\_type":grant\_type,  "client\_id":Key,  "client\_secret":Secret},  headers={"content\_type":content\_type,  "Api-Key":Key},  )  access\_token = req.json()["access\_token"]  return access\_token, Key  def get\_totalRoutes(data):  routes = data["\_embedded"]["routes"]  return len(routes)  def get\_routes(data):  routes = data["\_embedded"]["routes"]  return routes |