

Tools we wish we'd known more about as grad students

Daniel Cross and Robert Kosar

Github link to Robert's code:

https://github.com/r-kosar/MVDE_RKOSAR

What we hope you'll get out of this presentation.

Working with data is essential to the practice of statistics. Statistics as a field may give short shrift to practical knowledge. We hope this presentation will expose you to some practical tools that will help you in school and later.

In school:

- Speed up your research. Time you spend installing/munging is time you aren't spending doing more valuable things.
- Research exposure. It's a lot easier to be convinced a technique is useful if you can easily try it rather than having to implement it yourself. Exposing your research as an app is useful to other researchers. I still get inquiries about an [app of mine](#) half a decade later.
- Some of these tools can streamline tutorials you might give

After school:

- Data tends to get bigger, messier, and more collaborative after school, particularly in industry. Many tools that exist to help with these problems can be helpful during school.
- Having experience with these tools before you leave school puts you a step ahead of others!

Brief Survey

How many of you have experience with:

- Spark?
- Docker?
- Python?
- Ray?
- Elastic computing?
- Git/Github?
- Building an R package?
- Automated testing?
- Wikidata/any RDF?

About Robert

Rice Ph.D. Dec 2017

Lives in Santa Clara (SF Bay Area), CA

Data Scientist @eBay for ~4 years

I work on the search product building models/ testing features.















You have data-munging options!

There's more out there than dplyr!

- [Link to benchmarks](#)
- What questions should you ask before you get started?
 - Which tools will integrate most easily with my current workflow?
 - Do I have an eventual dataset size in mind?
 - Can I take advantage of distributed computing?
 - What will make it easy for my collaborators?
 - How fast does this need to be? Do I have an application in mind?
- How big can data be? Eventually you will need to do resort to distributed computing.

Component	Value
CPU model	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
CPU cores	40
RAM model	DIMM Synchronous 2133 MHz
RAM GB	125.78
GPU model	GeForce GTX 1080 Ti
GPU num	2

Input table: 100,000,000 rows x 9 columns (5 GB)

 DataFrames.jl	1.1.1	2021-05-15	9s
 Polars	0.8.8	2021-06-30	11s
 DuckDB	0.2.7	2021-06-15	14s
 data.table	1.14.1	2021-06-30	15s
 cuDF*	0.19.2	2021-05-31	17s
 ClickHouse	21.3.2.5	2021-05-12	18s
 spark	3.1.2	2021-05-31	34s
 pandas	1.2.5	2021-06-30	70s
 (py)datatable	1.0.0a0	2021-06-30	75s
 dask	2021.04.1	2021-05-09	170s
 dplyr	1.0.7	2021-06-20	175s
 Arrow	4.0.1	2021-05-31	212s

Running example for Robert's portion

The IRS requires publishes non-profit tax returns.

- ~500,000 non-profit returns / year
- Includes things like:
 - Assets
 - Directors
 - Highest-paid employees
 - etc.
- ~14 GB of XML data uncompressed
- Not quite 'big data' but processing speed /memory constraints come into the picture
- How can we speedily parse it?

name	title	total_comp (\$)
David Leebron	President	2264425
Allison Thacker	Chief Investment Officer, Treasurer, Assistant Secretary	1254590
Brent Smith	Associate Professor	991800
Peter Rodriguez	Dean of Jones Graduate Business School of Business	948917
Michael Bloomgren	Football Coach	926208
Carmen Thompson	Investment Manager	833274
Robert Yekovich	Dean of Shepherd School of Music (Jul - Aug)	808250
Reginald Desroches	Provost	781300
John Lawrence	Investment Manager	772854
Kevin Kirby	VP for Administration	656324

Aside: What's XML?

- Hierarchical data structure
- Designed to be both machine and human readable
 - Fails at both
 - hard for humans to read
 - slow for computers to parse
- Can add new data without screwing up the old data
- In principle, html is a subset of xml.

Example

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this
weekend!</body>
</note>
```

(from <https://www.w3schools.com/xml/note.xml>)

Parsing the IRS data (basic R function)

```
parse_person = function(person){
  ns=c(irs="http://www.irs.gov/efile")
  name = xml_find_first(person, ".//irs:BusinessName/irs:BusinessNameLine1Txt", ns=ns) %>% xml_text()
  comp = xml_find_first(person, ".//irs:ReportableCompFromOrgAmt", ns=ns) %>% xml_text()
  title = xml_find_first(person, ".//irs:TitleTxt", ns=ns) %>% xml_text()
  other = xml_find_first(person, ".//irs:OtherCompensationAmt", ns=ns) %>% xml_text()
  return(c(name=name, comp=comp, title=title, other=other))
}
parse_irs_xml = function(filename){
  ns=c(irs="http://www.irs.gov/efile")
  tryCatch(
    expr={
      my_xml = read_xml(paste0(path_to_data, 'xml/', filename))
      xml_find_first(my_xml, ".//irs:Filer/irs:BusinessName/irs:BusinessNameLine1Txt", ns=ns) %>%
        xml_text() -> institution_name
      xml_find_all(my_xml, ".//irs:Form990PartVIISectionAGrp", ns=ns) -> people
      out_list = lapply(people, parse_person)
      comp_tibble = bind_rows(out_list)
      comp_tibble = comp_tibble %>% mutate(total_comp = as.numeric(comp) + as.numeric(other))
      comp_tibble["institution"] = institution_name
      comp_tibble["filename"] = filename
      return(comp_tibble)
    },
    error=function(e){return(tibble())}
  )
}

files = list.files(paste0(path_to_data, 'xml/'))
s = system.time({
  parsed = lapply(files, parse_irs_xml)
})
```


How slow is it? (very)

Test Hardware:

2021 Macbook Pro

Apple M1 Pro 10 Cores

32 GB Ram

R loop

```
lapply(files, parse_irs_xml)
```

4228 seconds holy cow

python loop

```
[parse_irs_xml(filename) for filename in files]
```

351 seconds

How can we speed it up?

We could optimize the parsing function.

But it's easier to throw more hardware at it!

R 4228 seconds -> 2228 seconds

```
mclapply(files, parse_irs_xml, mc.cores=9)
```

Python 351 seconds -> 67 seconds

```
from joblib import Parallel, delayed
```

```
start_time = time.time()
```

```
parsed = Parallel(n_jobs=9)(delayed(parse_irs_xml)(f) for f in files)
```

Python ray 351 -> 125 seconds

```
@ray.remote
```

```
def parse_irs_xml_ray(filename):
```

Kinda slow... we will cheat to make ray competitive and do chunked Ray

Python Chunked ray 125 seconds -> 74 seconds

```
@ray.remote
```

```
def parse_irs_xml_ray_chunked(file_list):
```

Why is Ray awesome?

- Even though it was not the fastest option for this example, the cool thing about Ray is it can scale to multi-node clusters and still use the same basic code. So we could continue to speed things up by adding more computers.
- Allows you to easily scale your python code from 1 core to ??? cores.
- Ray clusters can accommodate special hardware like GPUs. You can use it to train the latest neural nets. OpenAI uses it for their models!
- It's cost-effective. Ray [Holds the cost efficiency record for sorting 100TB](#)
- More info about [Ray](#).

What about spark?

Spark is similar to Ray in that it can be used on anything from a pc to a big compute cluster to speed up data processing.

Pros:

- Very scaleable, petabyte+ possible
- Lots of integrations: pyspark, sparkR, sparklyR

Cons:

- Can be annoying to set up.
- May require a lot of hardware (use the cloud if you don't need to process all the time)
- Based on java/scala so making fundamental changes may be hard
- I frankly don't recommend spark unless you plan to scale beyond a single machine

Getting started with spark requires more installation, so first let's understand docker

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

–Leslie Lamport

Docker

A docker container is essentially a virtual computer. Why do you want a virtual computer?

- Sometimes it can be hard to install software. With docker, you can either find someone else who has already figured out how, or at least record how you do it in a way.
 - They're great for installing CUDA!
 - Complicated applications are easy to set up. For instance, you can use the selenium grid docker image to supercharge your web scraping!
 - Trying new software versions is safe and simple!
- The behavior of a docker image should be relatively similar wherever it's hosted. That makes it easy to share software with others because it should behave the same for them as it does for you!
 - This is great for tutorials. Everyone just needs to have docker installed and you don't have to worry about people being on different software versions.



Docker

We're going to use docker for two things:

1. Set up a spark cluster.
2. Create a docker client to use spark (this is optional, you could just install the necessary packages on your machine)

#1 is easy. Someone has already created this application which can be downloaded and installed thusly:

```
curl -L -o ./spark/docker-compose.yml
```

<https://raw.githubusercontent.com/bitnami/containers/main/bitnami/spark/docker-compose.yml>

```
docker-compose -f ./spark up
```

Docker compose basically lets you create applications involving multiple docker containers in an organized way.

This can be really useful for web testing. For instance, you can setup a selenium grid to test your web application with multiple browsers or to do web scraping in clever ways!

```
version: '3'
```

```
services:
```

```
spark:
```

```
image: docker.io/bitnami/spark:3.4.1
```

```
environment:
```

- SPARK_MODE=master
- SPARK_RPC_AUTHENTICATION_ENABLED=no
- SPARK_RPC_ENCRYPTION_ENABLED=no
- SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
- SPARK_SSL_ENABLED=no
- SPARK_USER=spark

```
ports:
```

- '8080:8080'
- '7077:7077'

```
networks:
```

- spark

```
volumes:
```

- type: bind
- source: ../r
- target: /app

```
spark-worker:
```

```
image: docker.io/bitnami/spark:3.4.1
```

```
environment:
```

- SPARK_MODE=worker
- SPARK_MASTER_URL=spark://spark:7077
- SPARK_WORKER_MEMORY=10G
- SPARK_WORKER_CORES=4
- SPARK_RPC_AUTHENTICATION_ENABLED=no
- SPARK_RPC_ENCRYPTION_ENABLED=no
- SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
- SPARK_SSL_ENABLED=no
- SPARK_USER=spark

```
networks:
```

- spark

```
volumes:
```

- type: bind
- source: ../r
- target: /app

```
networks:
```

```
spark:
```

```
driver: bridge
```

Check Spark is running

If that worked, you can navigate to localhost:8080 in your browser and examine your spark cluster!

Spark Master at spark://efa4df84ec27:7077

URL: spark://efa4df84ec27:7077
Alive Workers: 1
Cores in use: 4 Total, 0 Used
Memory in use: 10.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State
worker-20231022080812-172.19.0.3-39883	172.19.0.3:39883	ALIVE

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor
----------------	------	-------	---------------------	------------------------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor
----------------	------	-------	---------------------	------------------------

Docker Example

Almost all the docker work has been done for you by others

See

<https://journal.r-project.org/archive/2017/RJ-2017-065/RJ-2017-065.pdf> for more information on the rocker dockers (portable etc.)

And <https://eddelbuettel.github.io/r2u/> on why r2u is particularly good (it's fast basically)

These dockers are a great way to try out new packages without affecting your local R installation

```
# Use a base image that includes R
FROM rocker/r2u

# Install necessary packages
RUN apt-get update && \
    apt-get install -y openjdk-8-jdk scala wget && \
    apt-get clean

# Download and install Apache Spark 3.4.1
WORKDIR /opt
RUN wget
https://downloads.apache.org/spark/spark-3.4.1/spark-3.4.1-bin-hadoop3.tgz
&& \
    tar -xvzf spark-3.4.1-bin-hadoop3.tgz && \
    mv spark-3.4.1-bin-hadoop3.tgz spark

# Set environment variables
ENV SPARK_HOME=/opt/spark
ENV PATH=$SPARK_HOME/bin:$PATH

# Install R dependencies
RUN R -e "install.packages('tidyverse', dependencies=TRUE)"
# match your spark cluster version
RUN R -e "install.packages('sparklyR', dependencies=TRUE)"

# Set the working directory
WORKDIR /app
```


Using spark

Now that we have a spark cluster and a spark client, let's do some analysis!

To start the client (the cluster is already running)

```
docker run --network="host" -v ./r:/app -it my_docker
```

Load connect R to spark and load some data:

```
library(sparklyr)
spark_home="/opt/spark-3.4.1-bin-hadoop3"
config <- spark_config()
config$sparklyr.connect.enablehivesupport = FALSE
# this command may take a minute
sc <- spark_connect(spark_home=spark_home,
                    master = "spark://172.19.0.2:7077",
                    config=config)
data = spark_read_csv(sc, name = "credit_data",
                      path = "nonprofit_salaries.csv",
                      header = TRUE, delimiter = ",")
```

Find the address of your spark cluster

```
docker ps
```

```
CONTAINER ID
8e4ca77e6d28
efa4df84ec27
fa6ef95bd7e8
b3b468c20509
```

...

```
NAMES
spark-spark-worker-1
spark-spark-1
infallible_blackburn
stoic_mahavira
```

```
docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' spark-spark-1
```

```
172.19.0.2
```

I think it would be also possible to:

1. Set a static ip address so you don't have to look it up.
2. (better) Just launch you client as part of your docker-compose.

...

```
rclient:
image: my_docker
command: tail -F anything
networks:
- spark
```

Then set

```
master=spark://spark:7077
```

Using spark (cont'd)

```
fix_names = data %>%
  mutate(name=case_when(name=='NA' ~ NA,
                        TRUE ~ name)) %>%
  mutate(alt_name=case_when(alt_name=='NA' ~ NA,
                            TRUE ~ alt_name)) %>%
  mutate(derived_name = coalesce(name, alt_name))
%>%
  mutate(total_comp = as.numeric(total_comp))

fix_names %>%
  arrange(desc(total_comp)) %>%
  filter(row_number() <= 20) %>%
  print(width=200, n=12)
```

Ascension Health is a 'nonprofit' Catholic health network that has been criticized for its executives' pay [WSJ](#).

```
derived_name      title
<chr>             <chr>
1 LLOYD H DEAN    CHIEF EXECUTIVE OFFICER
2 JOHN M MURPHY MD PRESIDENT/CEO
3 MATTHEW GLINE   DIRECTOR AND SECRETARY
4 ERIC VENKER     DIRECTOR AND CHAIR
5 DR AMY GUTMANN  PRES.,EX-OFF TRUSTEE-THRU 3/22
6 DR AMY GUTMANN  PRES.,EX-OFF TRUSTEE-THRU 3/22
7 DANIEL MULLEN  HEAD COACH, FOOTBALL
8 ERNIE SADAU     PRESIDENT & CEO
9 Gary A Patterson Head Football Coach
10 KENNETH A SAMET CEO AND PRESIDENT
11 Daniel R Manning Former Men's BktBall Co
12 JOSEPH R IMPICCICHE JD MHA PRESIDENT & CEO
```

```
institution      total_comp
<chr>            <dbl>
1 DIGNITY HEALTH 35462873
2 WESTERN CONNECTICUT HEALTH NETWORK INC 30060915
3 ROIVANT SOCIAL VENTURES INC 27301494
4 ROIVANT SOCIAL VENTURES INC 23346834
5 TRUSTEES OF THE UNIVERSITY OF PENNSYLVANIA 22892227
6 TRUSTEES OF THE UNIVERSITY OF PENNSYLVANIA 22892227
7 UNIVERSITY ATHLETIC ASSOCIATION INC 18762075
8 Christus Health 17916470
9 TEXAS CHRISTIAN UNIVERSITY 17204401
10 MEDSTAR HEALTH INC 15867683
11 WAKE FOREST UNIVERSITY 14658459
12 Ascension Health Alliance 13707694
```

Using spark (cont'd)

```
fix_names %>%  
  filter(total_comp > 0) %>%  
  group_by(derived_name, total_comp) %>%  
  summarize(count = n()) %>%  
  mutate(total_total_comp = count * total_comp) %>%  
  arrange(desc(total_total_comp)) %>%  
  print(n=25)
```

	derived_name <chr>	total_comp <dbl>	count <dbl>	total_total_comp <dbl>
1	DR AMY GUTMANN	22892227	2	45784454
2	LLOYD H DEAN	9642249	4	38568996
3	LLOYD H DEAN	35462873	1	35462873
4	Timothy J Babineau MD	1293210	26	33623460
5	JOHN M MURPHY MD	30060915	1	30060915
6	MATTHEW GLINE	27301494	1	27301494
7	JOHN M MURPHY MD	4526905	6	27161430
8	JEFF PUCKETT	3578707	7	25050949
9	ERIC VENKER	23346834	1	23346834
10	Raymond P Vara Jr	2498142	8	19985136
11	DANIEL MULLEN	18762075	1	18762075
12	ERNIE SADAU	17916470	1	17916470
13	CRAIG A CORDOLA FACHE	1052538	17	17893146
14	ERNIE W SADAU	5921227	3	17763681
15	Gary A Patterson	17204401	1	17204401
16	KENNETH A SAMET	15867683	1	15867683
17	PETER AMMON	7380265	2	14760530
18	Daniel R Manning	14658459	1	14658459
19	DANIEL J WIDAWSKY	7050168	2	14100336
20	JOSEPH R IMPICICHE JD MHA	13707694	1	13707694
21	Jeffrey A Romoff UPM	12880205	1	12880205
22	JOSEPH R IMPICICHE JD MHA	2557835	5	12789175
23	JAVON R BEA PRESIDENT	12527853	1	12527853
24	ELLEN V FUTTER	11925437	1	11925437
25	ROBERT I GROSSMAN MD	11868034	1	11868034

Using spark (cont'd)

```
fix_names %>%  
  filter(derived_name == 'CRAIG A CORDOLA FACHE' &  
         total_comp == 1052538) %>%  
  pull(institution)
```

Alas, checking the paper returns this appears to just be a poorly named XML tag.

(D)	(E)	(F)
Reportable compensation from the organization (W-2/1099-MISC)	Reportable compensation from related organizations (W-2/1099-MISC)	Estimated amount of other compensation from the organization and related organizations

```
[1] "CMC FOUNDATION OF CENTRAL TEXAS"  
[2] "Ascension Texas"  
[3] "Tri-County Clinical"  
[4] "SETON HAYS FOUNDATION"  
[5] "PROVIDENCE FOUNDATION INC"  
[6] "ASCENSION SETON"  
[7] "Dell Children's Medical Group"  
[8] "SetonUT Dell Medical School University Physicians Group"  
[9] "THE SETON COVE"  
[10] "ALEXIAN BROTHERS HEALTH SYSTEM"  
[11] "Seton Family of Doctors"  
[12] "TWENTY-SIX DOORS INC"  
[13] "Seton Family of Pediatric Surgeons"  
[14] "SETON ORAL & MAXILLOFACIAL SURGERY"  
[15] "SETON WILLIAMSON FOUNDATION"  
[16] "Blue Ladies Minerals Inc"  
[17] "Ascension Texas Cardiovascular (fka Ascension Texas Heart & Vascular Instit"
```

```
<ReportableCompFromOrgAmt>0</ReportableCompFromOrgAmt>  
<ReportableCompFromRltdOrgAmt>4633536</ReportableCompFromRltdOrgAmt>  
<OtherCompensationAmt>1052538</OtherCompensationAmt>
```

More Docker Resources

[Docker Overview](#)

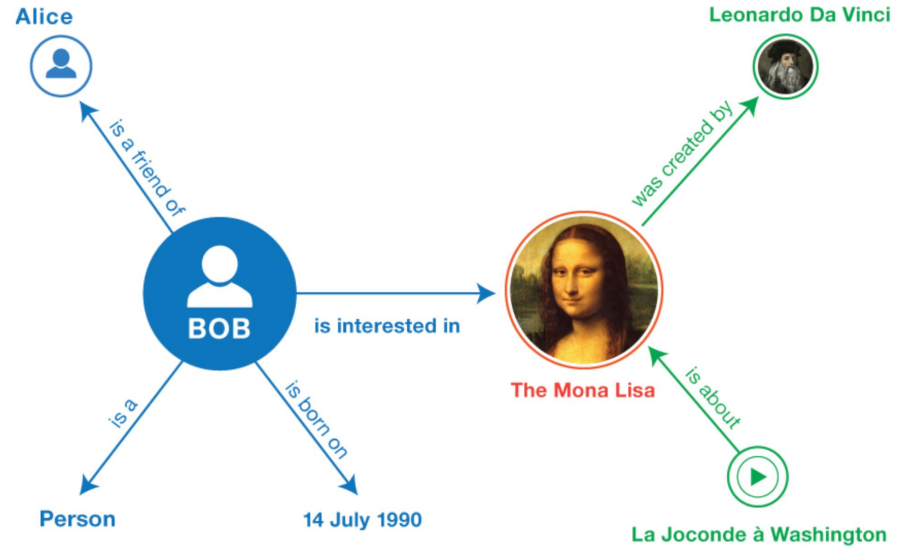
[Install Docker Desktop](#) (free for individuals)

Chat GPT is pretty good at building images for you!

Consider Using RDF!

RDF represents data as a set of subject-predicate-object triples

Allows for very flexible data relationships



From <https://www.w3.org/TR/rdf11-primer/>

Wikidata has tons of info!

[Rice University - Wikidata](#)

- 15B+ triples
- Great way to annotate your data.

```
SELECT DISTINCT ?uni ?uniLabel ?address WHERE {  
  ?uni wdt:P31/wdt:P279* wd:Q3918.  
  ?uni wdt:P6375 ?address.  
  SERVICE wikibase:label {  
    bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en".  
  }  
}
```

uni	Q wd:Q270272
uniLabel	The Rockefeller University
address	1230 York Avenue, New York, NY, 10065-6399

uni	Q wd:Q49205
uniLabel	Wellesley College
address	106 Central St, Wellesley, MA, 02481-8203

uni	Q wd:Q519427
uniLabel	University of Denver
address	2199 S University Blvd, Denver, CO 80208

[query link](#)

Bonus Section: What should I know if I'm thinking about getting a job in big tech?

Big tech data job breakdown

Data Scientist:

- Most general position, jack-of-all-trades
- Actual job will vary by company
- Most natural position for statistician
- Pay \$\$-\$\$\$\$

Data Analyst

- Bridge between businesspeople and engineers
- Careful, sometimes just SQL
- Pay \$-\$\$

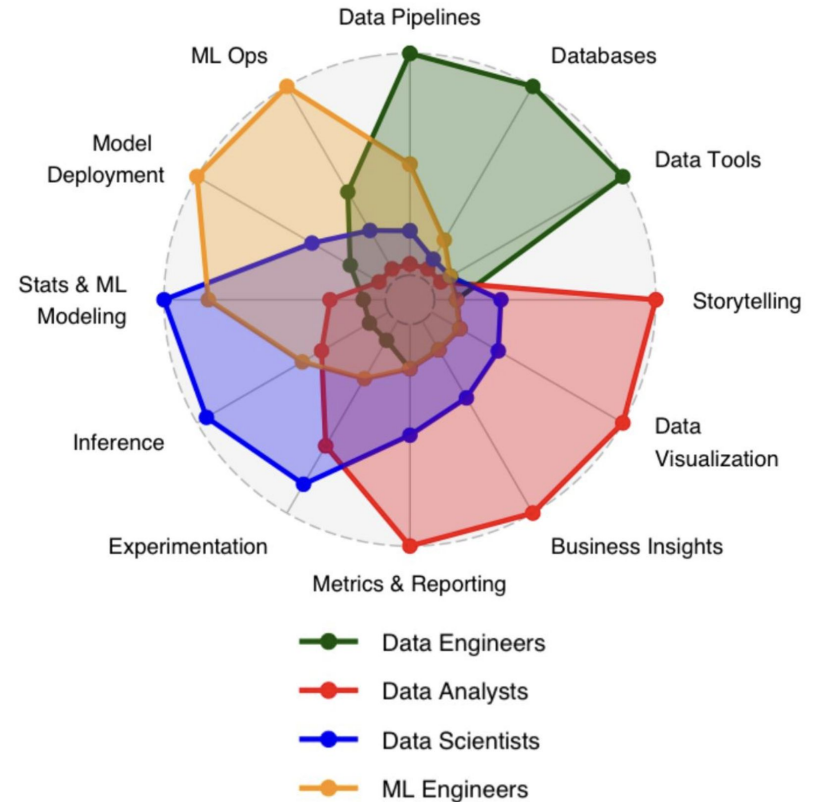
Data Engineer

- Software engineer that specializes in data
- More of an engineering discipline than a stats discipline.
- Pay \$\$\$

ML Engineer

- Software engineer that focuses on machine learning
- Possible for statisticians to transition with a little CS prep
- Pay \$\$\$-\$\$\$\$

\$-ranges are within a given level. You may be more likely to reach a higher level for something you enjoy!



Graphic from

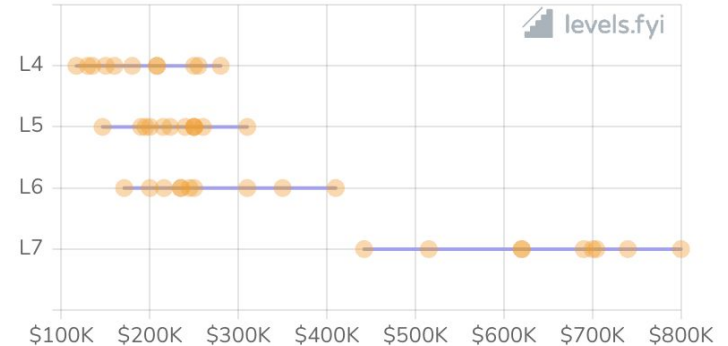
<https://www.hibernian-recruitment.com/whats-a-data-scientist-explaining-roles-in-big-data/>

(but I don't think their idea)

Big Tech Must-Have resources

- The best site for big tech companies is
 - <https://www.levels.fyi>
- Example for Amazon data scientist pictured right.
- Tech interviews tend to include the following:
 - Coding interview
 - SQL
 - Algorithms
 - 'Behavioral' Interviews
 - Tell me about a time you were in some situation you're about to lie about.
 - 'Methods' Interviews
 - How would you test some product change
 - Math Interviews
 - Compute some probability
- The coding interviews have been gamified to the point where it's hard to pass without preparation. See <https://leetcode.com/problemset/all/>

Salary Range Chart



Trajectory Chart



<https://www.levels.fyi>

Job application Wisdom

1. For large companies, there is usually information out there about the interview process. Check reddit/blind/etc.
2. Don't hesitate to reach out to weak connections for referrals to big tech companies, there's often no downside to providing a referral to someone, only upside.
3. Don't take rejection personally. Often the reason is ridiculous.

Miscellaneous Advice

When creating an application for your research/project, try to maintain as much control as possible.

- When providing a link to some application, use a link shortener so you can change the host later if need be.

Questions?

About Daniel

Energy sector Data Scientist for Shell E&P, retail energy providers, and vendors

Works on creating and maintaining models for oil well planning and load/price forecasting

- Hedge and price optimization

- Weather hedging

Life in Energy Industry / 'Small' Tech

SaaS vs Retail Energy Provider vs Shell E&P

Smaller company means less red-tape

Work from home is prevalent

- Requires home workspace

Server resources may be less/harder to get

Data scientists may need to work in areas not in job description

May be only employee with significant statistical knowledge

SME on sales/customer calls

Tight timelines for new products/improvements

Life in Energy Industry / 'Small' Tech Cont.

Seasonal workloads

Time of day dependant

Driven by Eastern and Central time

Mix of R&D and operations

Work closely with traders, finance, and pricing

Regulated vs Deregulated markets

Third party results

L2E Partial Mixture Modelling

L2E Partial Mixture Estimation

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\int f(x|\theta)^2 dx - \frac{2}{n} \sum_{i=1}^n f(x_i|\theta) \right]$$

L2E Partial Mixture Modelling Cont.

Normal model often used

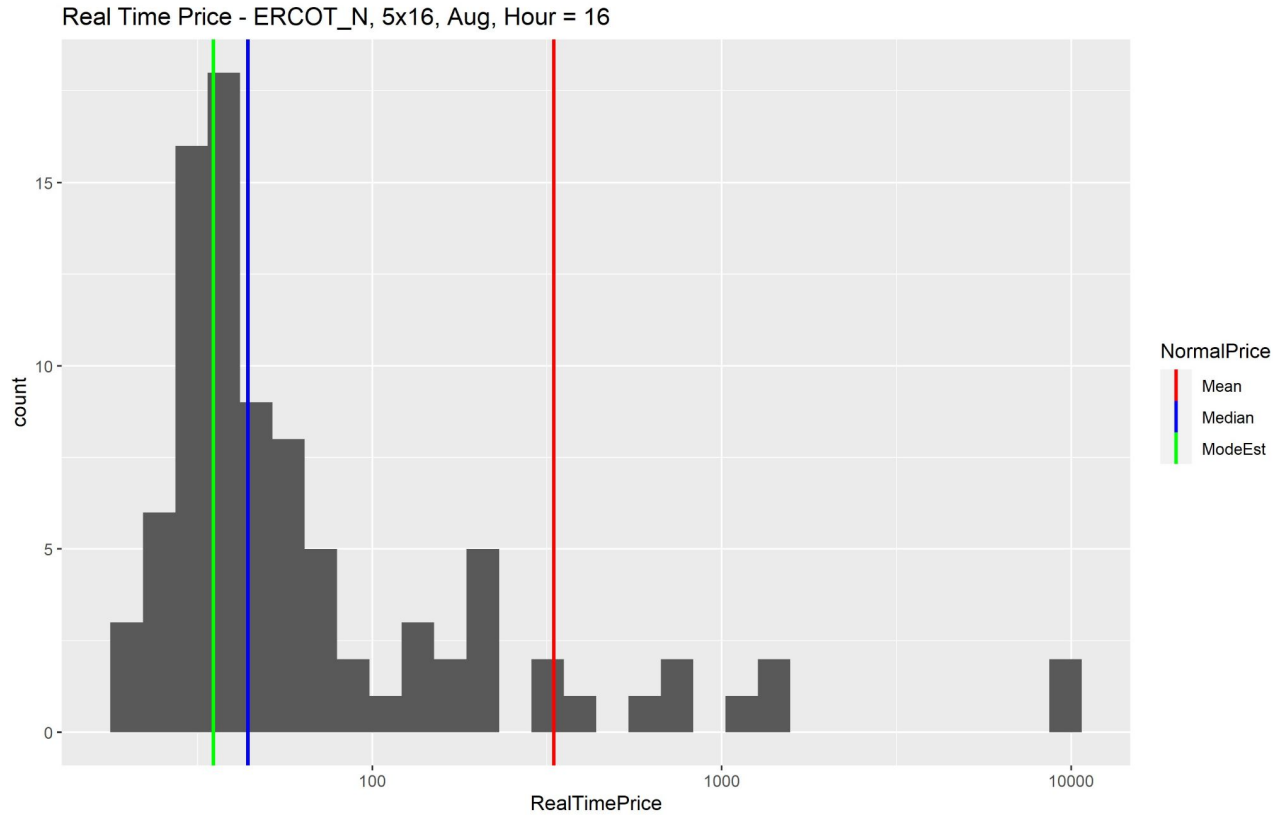
Symmetric, smooth, etc

Useful for “messy” data

Can be used in many applications

Downside: Can be slow to minimize

ERCOT Price Example



dplyr Pipelines

Medium speed

Easy to code and understand

Operator '>%>' has a shortcut in RStudio (Ctrl + Shift + M)

Can be combined with purrr and other packages easily

'do' function is versatile

Robust and easy to parse error/warning messages

SQL Integration with R

SQL is widely used in industry

Easy to learn hard to master

'rodbc' package and others

You can simply copy and paste most queries into R with minimal alterations

Easy to pull data from SQL/postgresql servers

Easy to insert data as well

Set nocount to on if using temp tables

SQL Integration with R Example

Simple to use:

```
stationMap = sqlQuery(  
  channel = riskProd,  
  query = paste0(  
    "select * from weather.dbo.UtilityweatherMap  
  where  
  zone in ('",paste(unique(histLP$Zone), collapse = "','"),"')  
  and ISO = '",  
    ISO,  
    "'  
  "  
  ),  
  stringsAsFactors = F  
) %>%  
as_tibble() %>% rename(stationCode = code, stationweight = share)
```

Integration With R Cont.

Rcpp for C++

Reticulate for Python

R.matlab for MATLAB

R Package Creation

Useful for sharing functions in professional environment

Help texts are useful to others using content

Simple to create and update

Make Vignettes (package descriptions) to help other users/yourself

Code organization

Consistent documentation

Code distribution

R Package Creation Cont.

Package “devtools” and “roxygen” useful

Create one or more R files with functions (grouping by type may be beneficial)

Do NOT use ‘library’ or ‘require’ functions - may cause conflicts

Use ‘::’ instead

Use #’ to comment your functions’ help texts

Use ‘use_data’ to include datasets

Advice

When creating a product make sure to return descriptive error messages

Current and future users may not be well versed in the language

Ask more experienced coworkers for advice and explanations

Try to get references from in-sector workers if possible

Use interviews for suboptimal jobs as practice if needed

LaTeX for resume - avoid Word

Be customer focused (internal or external)

Pad ETAs to account for emergencies that may arise