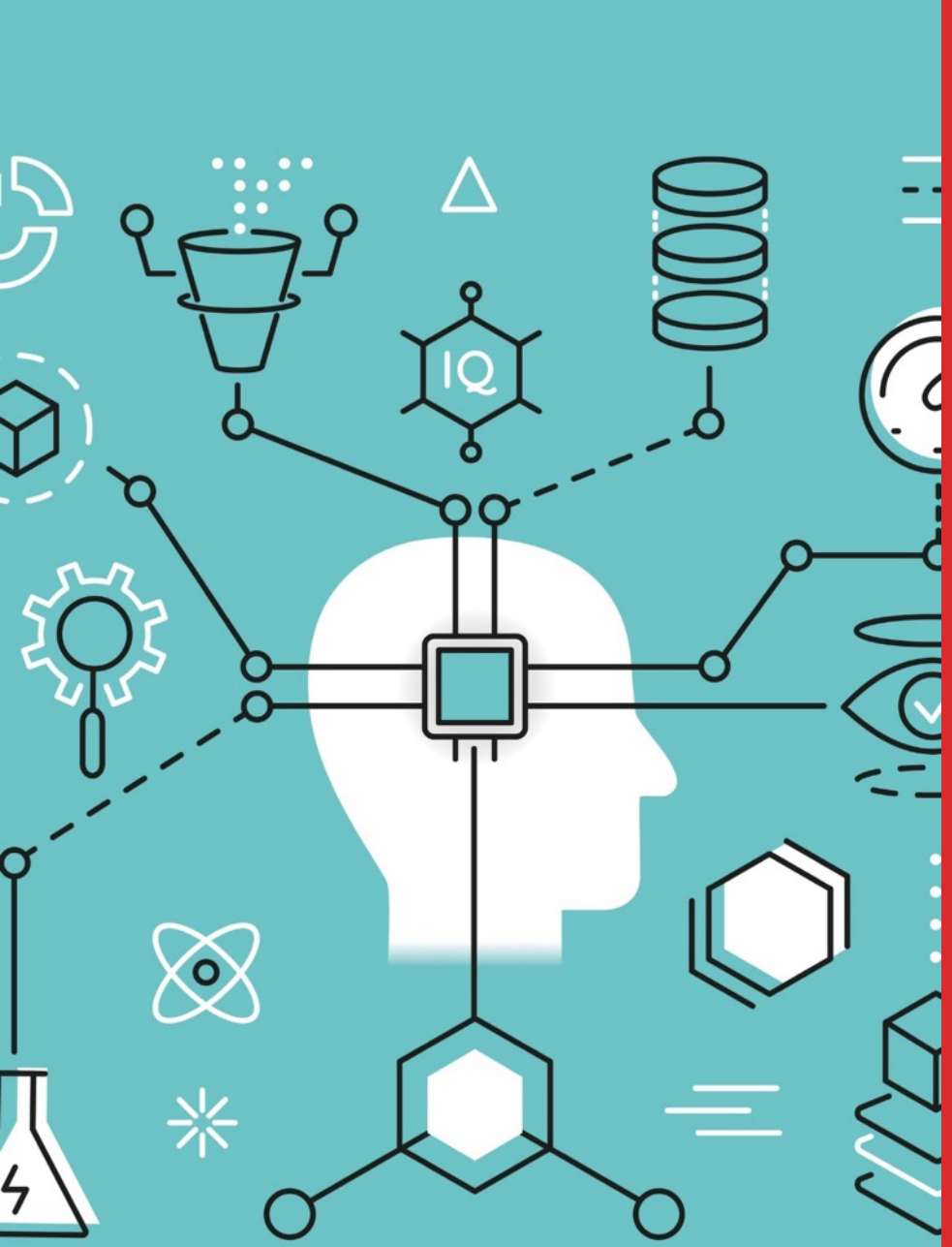
2019 Predictive Analytics Symposium

Session 33: B/I - Languages of Predictive Analytics (PA): A Tower of Babel?

SOA Antitrust Compliance Guidelines
SOA Presentation Disclaimer



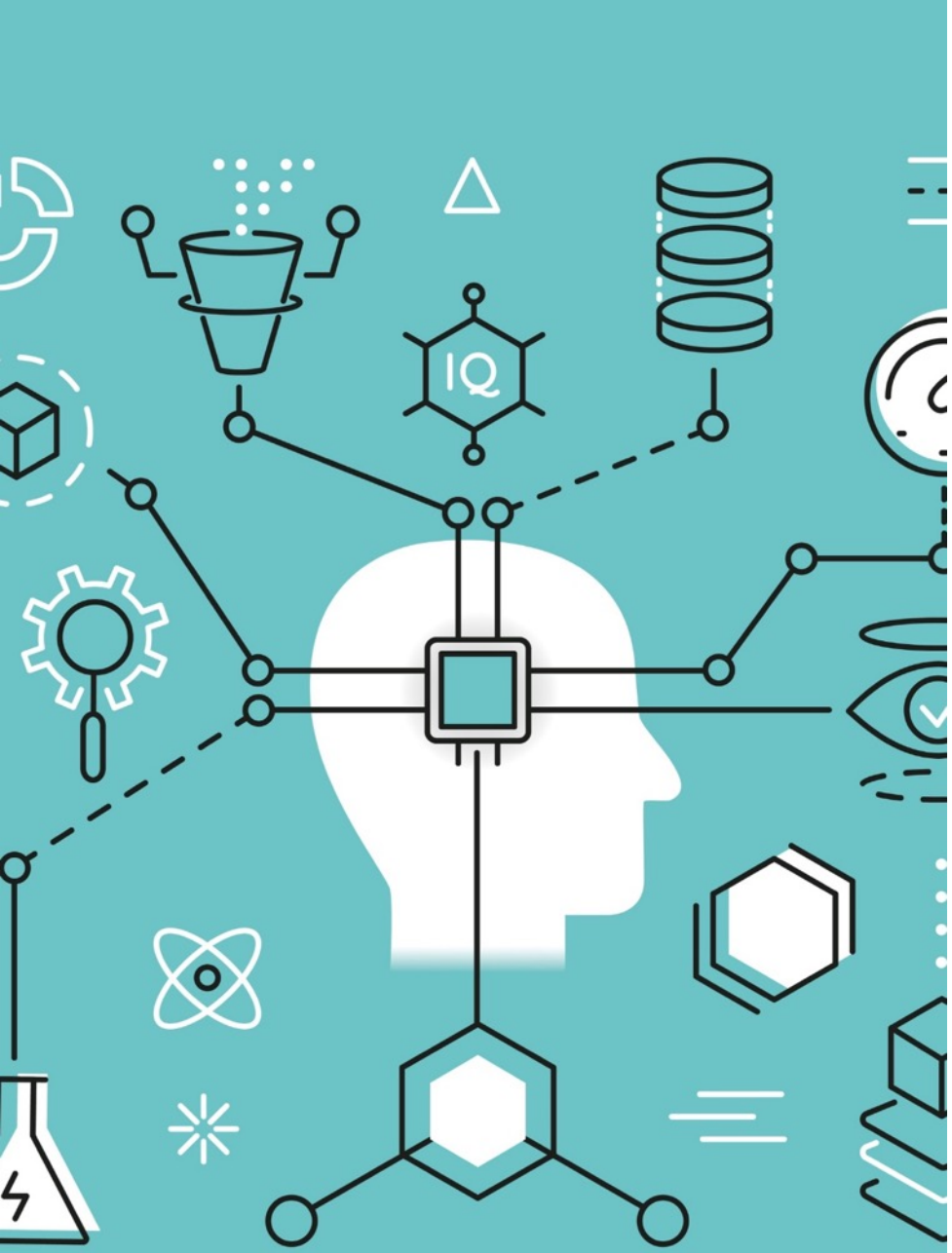
RGA

Session 33: Languages of Predictive Analytics: A Tower of Babel?

Jeff Heaton, Ph.D., FLMI

VP, Data Science, RGA Global Research & Data Analytics

09-20-2019



RGA

The views expressed in this presentation are my own and do not necessarily reflect the views of the Society of Actuaries (SOA), nor the views of Reinsurance Group of America (RGA).

Programming Languages

An electronic tower of Babel?

- Julia
- MatLab/Octave
- Python
- R
- SAS
- PySpark
- Scala

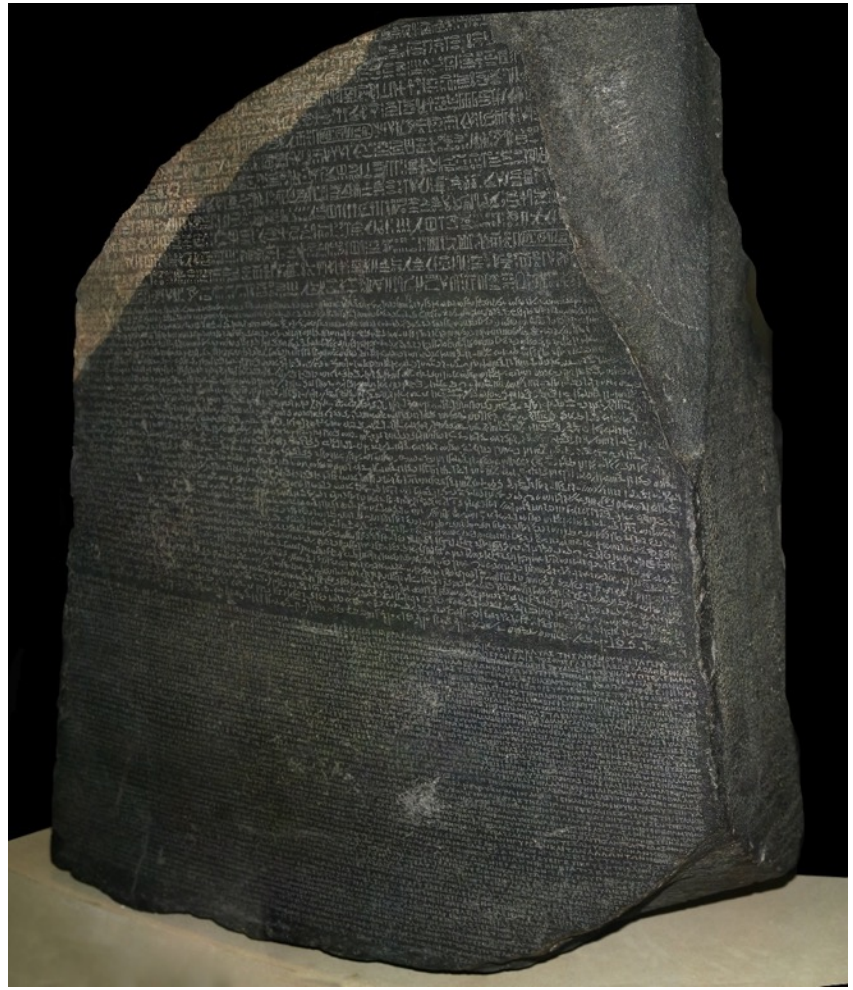
The Tower of Babel by Pieter Bruegel the Elder(1563)

RGA

What is a Programming Language?

Are they the same as human languages?

- Formal Language
- Natural Language



The Rosetta Stone (Wikipedia)

RGA

Types of Programming Language

General purpose languages vs domain specific languages

- Domain Specific Programming Language (DSL):
  - MATLAB/Octave
  - PySpark
  - R
  - SAS
  - SQL
- General Purpose Programming Language:
  - Julia
  - Python
  - Scala

RGA

Sapir-Whorf Hypothesis

Linguistic Relativity

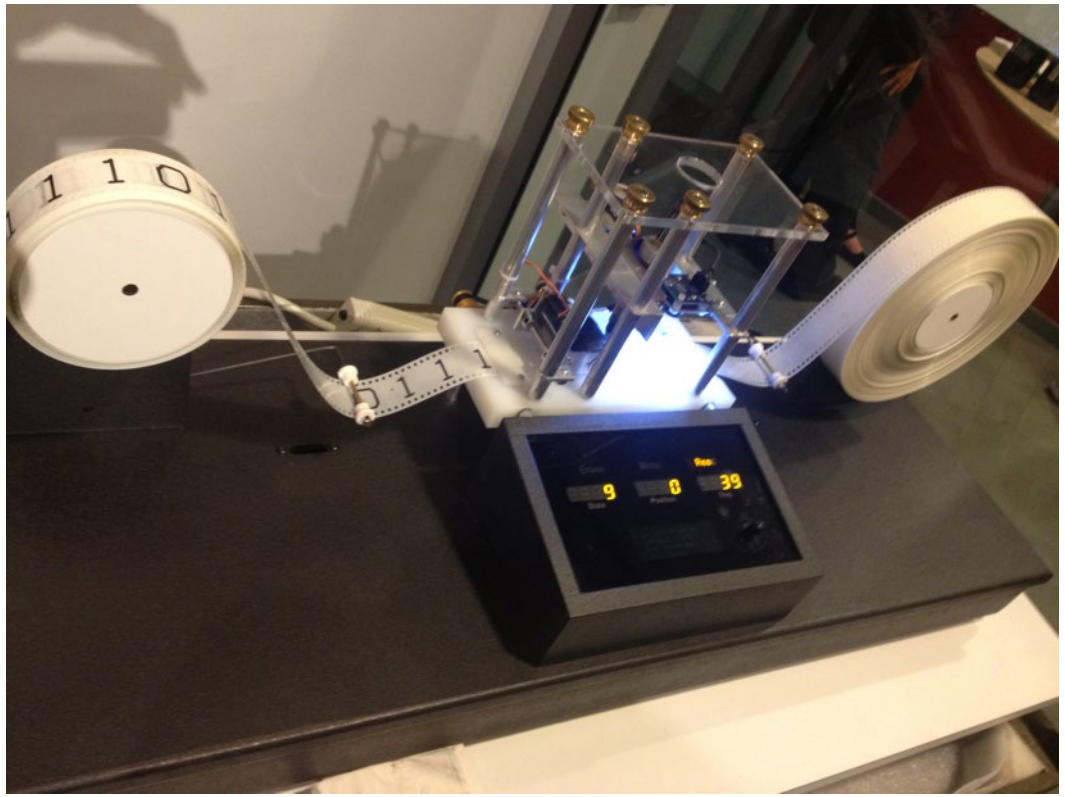
- Theory developed by Edward **Sapir** and Benjamin Lee **Whorf**.
- The **Sapir-Whorf hypothesis** proposes that the structure of a language determines or greatly influences the modes of thought and behavior characteristic of the culture in which it is spoken.
- Is one natural language better suited to a particular task?
- Is one programming language better suited to a particular task?

RGA

Turing Completeness

Are all programming languages the same?

- Turing's famous paper: "On Computable Numbers, with an Application to the Entscheidungsproblem"
- Is a programming language Turing complete?
- Has the core concept of programming languages changed since 1936?
- What differences are just "syntax sugar"?



Turing Machine, from Wikipedia, https://en.wikipedia.org/wiki/Turing_machine

RGA

Raw Performance

Compare mathematical performance of several languages:

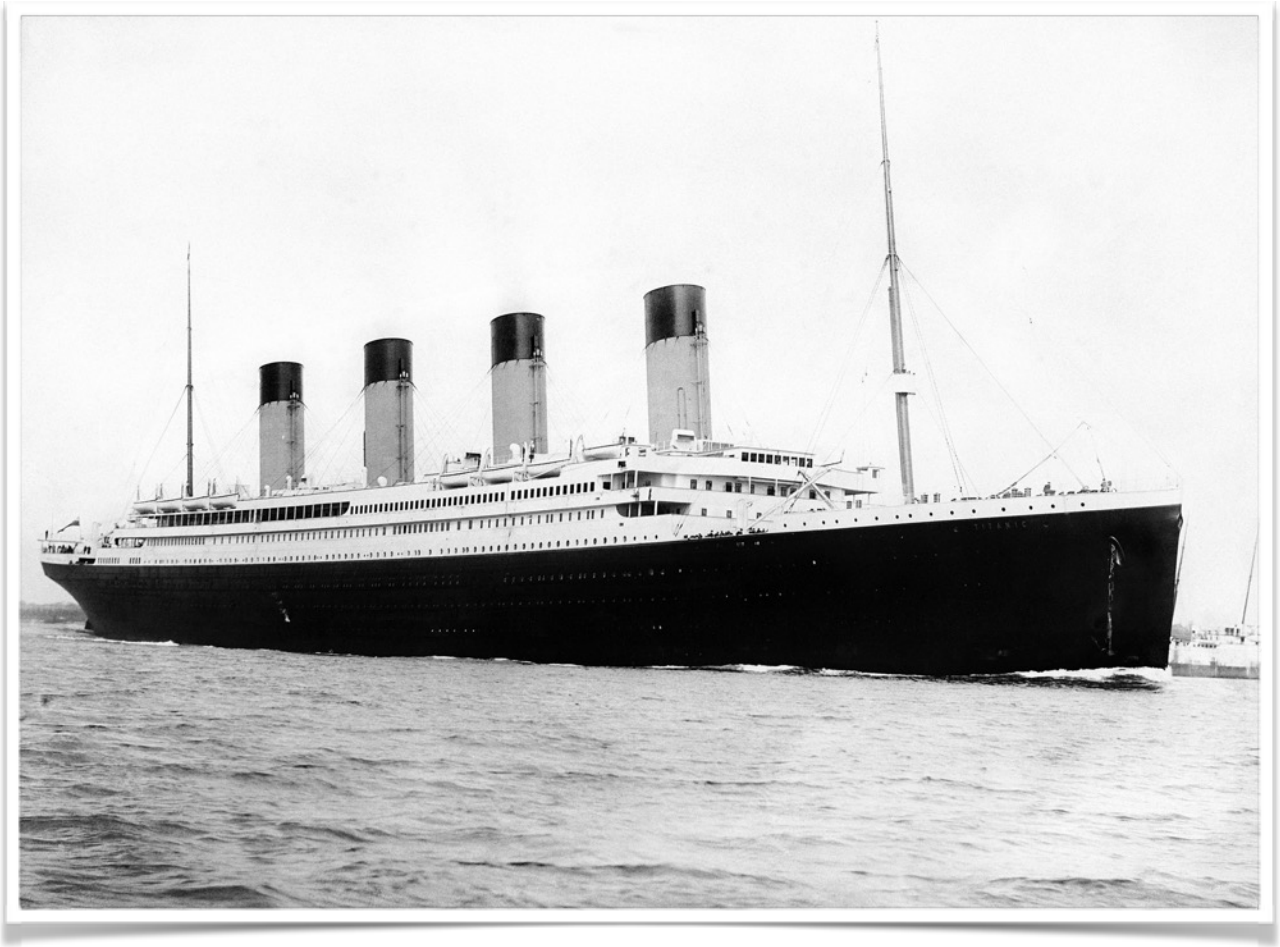
	pure python	c	numba	numpy	cython	scipy	lapack	julia
5	0.000051	0.000016	0.000002	0.000074	0.000006	0.000029	0.000031	6.091400e-07
10	0.000312	0.000016	0.000003	0.000234	0.000006	0.000030	0.000031	1.060710e-06
30	0.007800	0.000028	0.000014	0.001950	0.000014	0.000070	0.000056	9.082080e-06
100	0.289310	0.000154	0.000463	0.029782	0.000309	0.000309	0.000309	2.265530e-04
200	2.277604	0.001800	0.007200	0.119600	0.003600	0.001200	0.001200	1.740604e-03
300	7.636214	0.007800	0.019500	0.226200	0.007800	0.003900	0.001300	5.823171e-03
400	18.267632	0.017829	0.051257	0.514801	0.020057	0.008914	0.002229	1.372135e-02
600	62.197309	0.062400	0.124800	0.982802	0.088400	0.036400	0.010400	4.543215e-02
1000	290.472510	0.257401	0.569401	3.042005	0.288600	0.070200	0.039000	2.642414e-01

IBM DeveloperWorks, https://ibm.co/20XSZ4F

RGA

Titanic Tutorial Challenge

- Practice Kaggle competition
  - No ranking points, but lots of fun!
- Predict mortality – did passenger:
  - Survive?
  - Perish?
- From passenger features:
  - Gender
  - Name
  - Passenger class
  - Age
  - Family members present
  - Port of embarkation
  - Cabin
  - Ticket



https://www.kaggle.com/c/titanic

RGA

Learning from the Titanic Data

Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	3	Owen Harris	male	22	1	0	A/5 21171	7.25		S
1	1	riggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
1	3	ı, Miss. Laina	female	26	0	0	O2. 3101282	7.925		S
1	1	ily May Peel)	female	35	1	0	113803	53.1	C123	S
0	3	/illiam Henry	male	35	0	0	373450	8.05		S
0	3	n, Mr. James	male		0	0	330877	8.4583		Q
0	1	Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
0	3	ɔsta Leonard	male	2	3	1	349909	21.075		S
1	3	elmina Berg)	female	27	0	2	347742	11.1333		S
1	2	dele Achem)	female	14	1	0	237736	30.0708		C
1	3	ırguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
1	1	iss. Elizabeth	female	58	0	0	113783	26.55	C103	S
0	3	/illiam Henry	male	20	0	0	A/5. 2151	8.05		S
0	3	∧nders Johan	male	39	1	5	347082	31.275		S
0	3	nda Adolfina	female	14	0	0	350406	7.8542		S

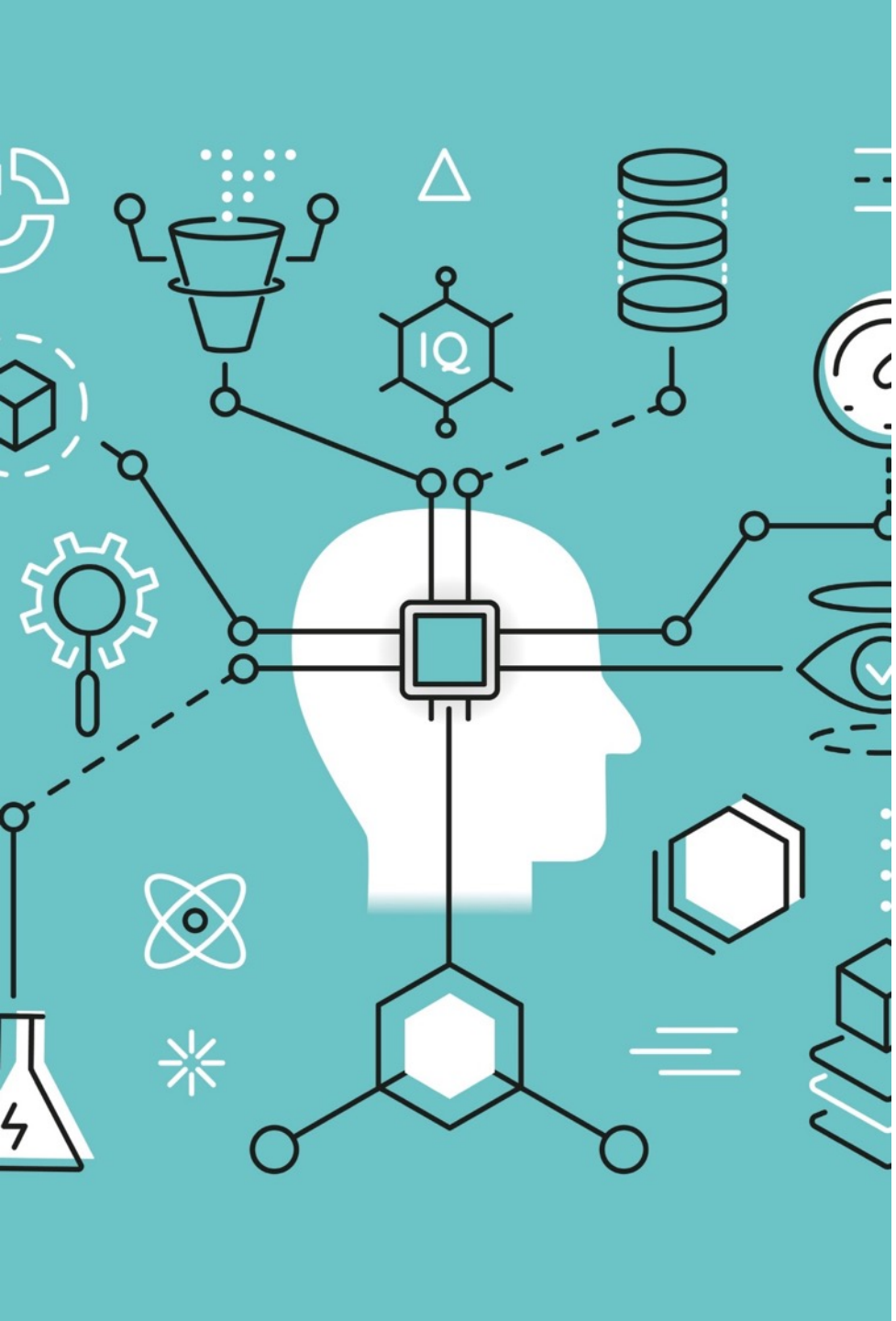
RGA

Code Availability

All code is available on Jeff Heaton's GitHub page:

https://github.com/jeffheaton

RGA



RGA

Julia

Open source programming language introduced by Jeff Bezanson (and others) in 2012 while they were at MIT. Julia is designed for numerical computing.

Reasons to Use Julia

The functionality and ease of use of R and Python with the speed of Java and C++.

- Direct calling of low-level C/C++ and Fortran code.
- Best-of-breed open source C and Fortran libraries for linear algebra, random number generation, signal processing, and string processing.
- Optional typing: Why choose between static and dynamic typing?
- Built-in, extensive multiprocessing and distributed parallel execution capabilities based on messaging.
- Julia Box: https://www.juliabox.com/

RGA

Reasons Not to Use Julia

Bleeding edge language with growing community and package support.

- Somewhat rare programming language; difficult to find programmers.
- Very cutting-edge, growing (but relatively small) community.
- Somewhat confusing type-system (separate types for NAs).
- Steeper learning curve than languages such as Python or Excel.
- Some questions of performance vs. other languages.

http://zverovich.net/2016/05/13/giving-up-on-julia.html

RGA

Julia Example

Read and Preprocess Data

```
df = readtable("./data/titanic-dataset.csv");

delete!(df, :PassengerId);
delete!(df, :Name);
delete!(df, :Ticket);
delete!(df, :Cabin);
df[isna.(df[:Age]),:Age] = median(df[ .~isna.(df[:Age]),:Age])
df[isna.(df[:Embarked]),:Embarked] = "S"
pool!(df, [:Sex]);
pool!(df, [:Embarked]);
```

RGA

Julia Example

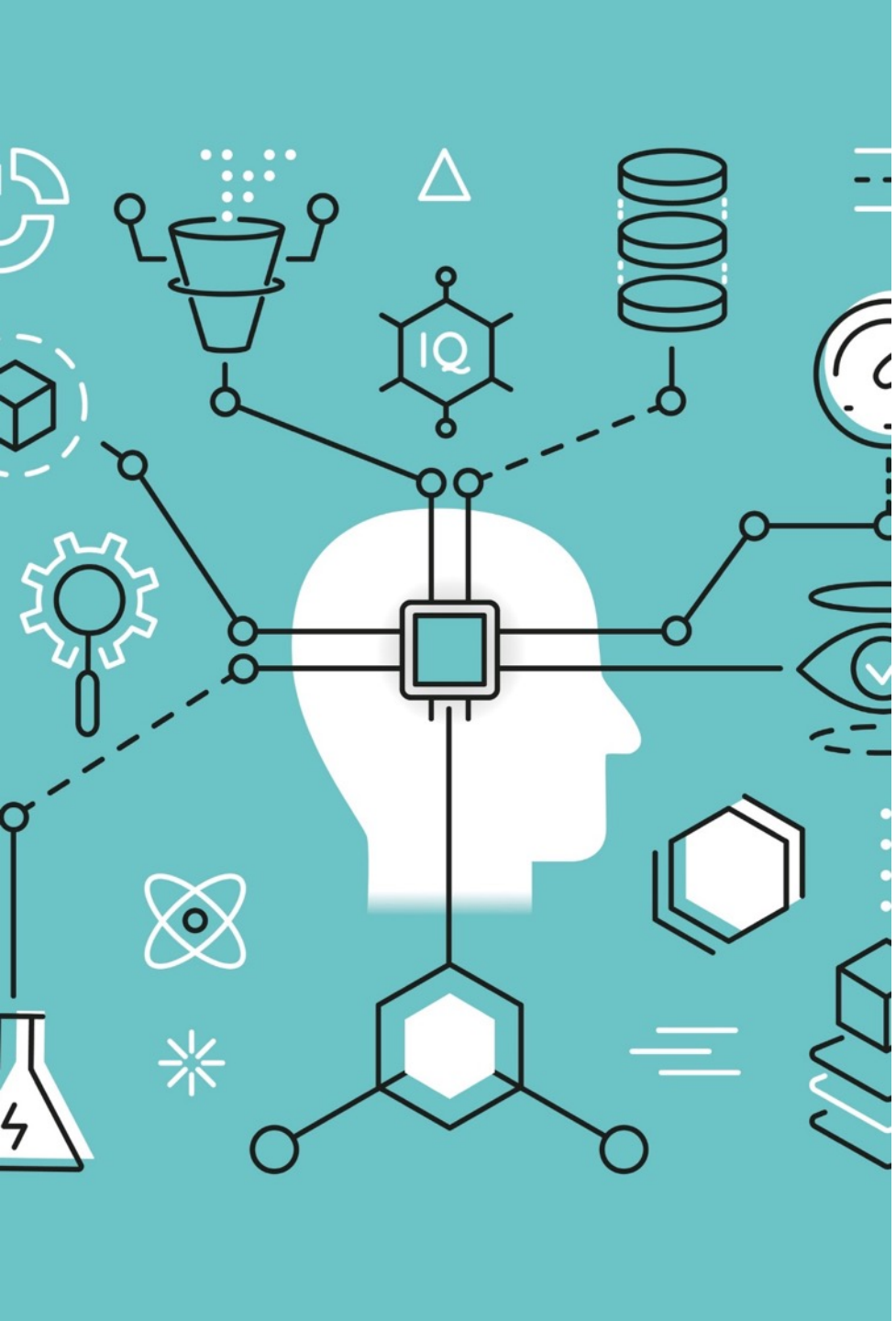
Training and Validation Split

```
split_pt = trunc(Int,size(df,1)*0.7) # 70% validation
shuffle_idx = sample(1:size(df,1),size(df,1));
df_train = df[1:split_pt,:];
df_validate = df[split_pt+1:size(df,1),:];
```

Julia Example

Fit Logistic Regression and Predict

```
model = glm(@formula(Survived ~ Pclass + Sex + Age + SibSp + Parch +
Fare + Embarked), df_train, Binomial(), LogitLink());

pred = predict(model,df_validate);
pred = convert(DataArray{Int}, round.(pred));

print("Accuracy: ")
println( sum(pred .== df_validate[:Survived]) / length(pred))
```



RGA

GNU Octave/MATLAB

GNU Octave is an open source programming language released in 1988 for numerical programming. GNU Octave is a free alternative to MATLAB.

MATLAB (matrix laboratory) is a commercial programming language introduced by MathWorks in 1984 for numerical computation.

Reasons to Use MATLAB

Advanced numerical programming language with matrix mathematics at its heart.

- Basic data element is the matrix. A simple integer is considered a matrix of one row and one column.
- Advanced graphing. Python's most popular graphing library (Matplotlib) is based on MATLAB.
- Quick prototypes of machine learning applications.
- Extensive toolbox (library) support for very specialized tasks.
- Simulink - a graphical programming environment for modeling, simulating and analyzing multi-domain dynamic systems.

RGA

Reasons Not to Use MATLAB

Expensive proprietary language without intrinsic support of multiprocessing.

- Expensive – a single copy of MATLAB is $2K-$8K, toolkits cost extra, and everyone needs a license.
- Octave provides some features of MATLAB for free, but compatibility is incomplete.
- MATLAB has performance issues for any code that is not specifically written as matrix operations.
- Steeper learning curve than languages such as Python or Excel.

RGA

MATLAB Example

Read and Preprocess Data

```
% Load the data
ds = readtable('titanic-dataset.csv');

% Handle missing ages
ds.Age(isnan(ds.Age)) = nanmean(ds.Age);

% Handle categoricals
ds.Embarked = categorical(ds.Embarked);
t = dummyvar(categorical(ds.Sex));
ds.Sex = t(:,1);
```

RGA

MATLAB Example

Split X&Y

```
% Split X & Y.
y = ds(:,'Survived');
x = ds(:,{'Pclass','Sex','Age','SibSp','Parch','Fare'});

% Create training matrix (all numeric)
x = table2array(x);
x = horzcat(x,dummyvar(ds.Embarked));
y = table2array(y);
```

MATLAB Example

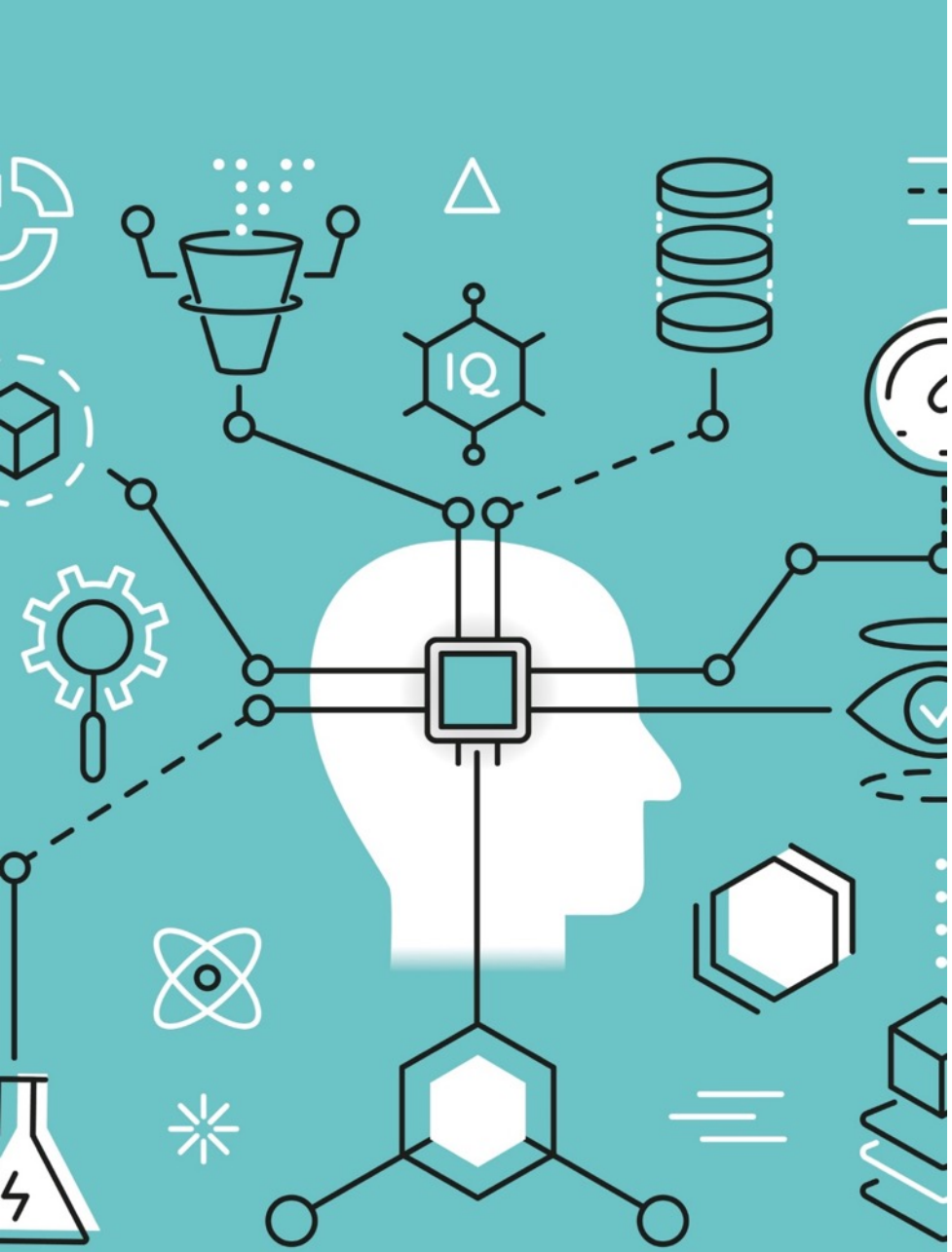
Training and Validation Split

```
% Training & validation split
[trainInd,valInd] = divideblock(length(x),0.7,0.3);
x_train = x(trainInd,:);
y_train = y(trainInd,:);
x_val = x(valInd,:);
y_val = y(valInd,:);
```

MATLAB Example

Fit Logistic Regression and Predict

```
% Fit the model
model = glmfit(x_train,y_train,'binomial','link','logit');

% Predict and calculate accuracy.
pred = glmval(model,x_val,'logit');
pred = round(pred);
acc = (pred == y_val);
sum(acc)/length(acc)
```



RGA

Python

Open source programming language introduced by Guido van Rossum in 1991 as a general purpose programming language.

Reasons to Use Python

Great mix of programming, charting, and user interface elements.

- Type-less programming language.
- TensorFlow, Numpy, and Scipy provide highly optimized distributed linear algebra operations for Python.
- Large community and add-on libraries.
- Extensive array of machine learning, statistical, and artificial intelligence libraries.
- Shallow learning curve, with a syntax that often requires much less formality than other languages.

RGA

Reasons Not to Use Python

Great “glue” language, but slow when used in pure form.

- Type-less programming language. (Yes, this is a pro and a con.)
- Nested loops will be slow unless there is a library, such as Numpy, Scipy or TensorFlow to do the heavy lifting.
- Libraries may not always have out-of-the-box support for Windows or Mac.
- Python 2 vs. Python 3.
- Programmers will either love or hate the whitespace scoping/blocking.
- Not commercially supported like Excel, SAS, or MATLAB.

RGA

Python Example

Read Data

```
path = "./data/"

def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])
    for x in dummies.columns:
        dummy_name = "{}-{}".format(name, x)
        df[dummy_name] = dummies[x]
    df.drop(name, axis=1, inplace=True)

filename_read = os.path.join(path,"titanic-dataset.csv")
df = pd.read_csv(filename_read,na_values=['NA','?'])
```

Python Example

Preprocess Data

```
df.drop('Name',1,inplace=True)
df.drop('PassengerId',1,inplace=True)
df.drop('Ticket',1,inplace=True)
df.drop('Cabin',1,inplace=True)
df['Sex'].replace('female', 0,inplace=True)
df['Sex'].replace('male', 1,inplace=True)
med = df['Age'].median()
df['Age'].fillna(med,inplace=True)
df['Embarked'].fillna('S',inplace=True)
encode_text_dummy(df,'Embarked')
```

Python Example

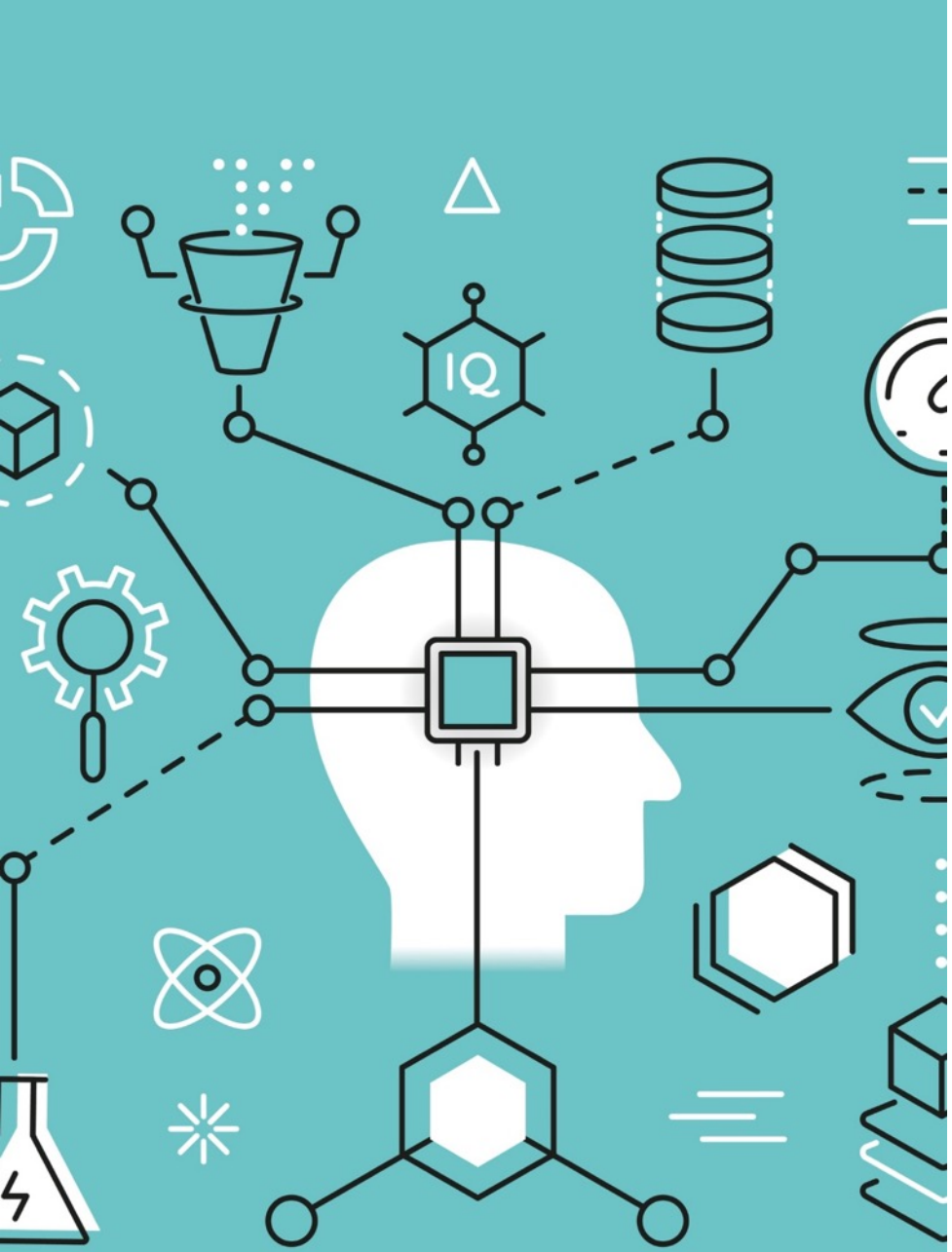
Split X&Y, Validation Train Split

```
x =
df.as_matrix(['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked-
C','Embarked-Q','Embarked-S'])
y = np.ravel(df.as_matrix(['Survived']))

x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.25, random_state=42)
```

Python Example

Fit Logistic Regression and Predict

```
classifier = LogisticRegression()

classifier.fit(x_train,y_train)

pred = classifier.predict(x_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy score: {}".format(score))
```



RGA

R

Open source programming language introduced in 1993 by the R Programming Team. R is a general purpose programming language primarily designed for statistical analysis.

Reasons to Use R

Open source programming language for statistical computing and graphics.

- Support for a vast array of statistical techniques.
- Statisticians who develop new methods often work in R, so R users often get to use them immediately.
- KDD Nuggets rates R as the software language most commonly used for data science.

http://www.kdnuggets.com/polls/2015/analytics-data-mining-data-science-software-used.html

RGA

Reasons Not to Use R

Steep learning curve and very different programing paradigm than other languages.

- R is a strange, deeply flawed language that nevertheless has an enthusiastic and rapidly growing user base. (John Cook, MSDN)
- Everything must fit into RAM (usually).
- Larger learning curve than Excel or Python.
- Not commercially supported like Excel, SAS, or MATLAB.

https://channel9.msdn.com/Events/Lang-NEXT/Lang-NEXT-2012/Why-and-How-People-Use-R

RGA

R Example

Load Data and Preprocess

```
df = read.csv("./data/titanic-dataset.csv", header = TRUE)

drops <- c("PassengerId","Name", "Ticket", "Cabin")
df <- df[ , !(names(df) %in% drops)]
df$Age[is.na(df$Age)] <- median(df$Age, na.rm=TRUE)
df$Embarked [is.na(df$Embarked)] <- 'S'
```

R Example

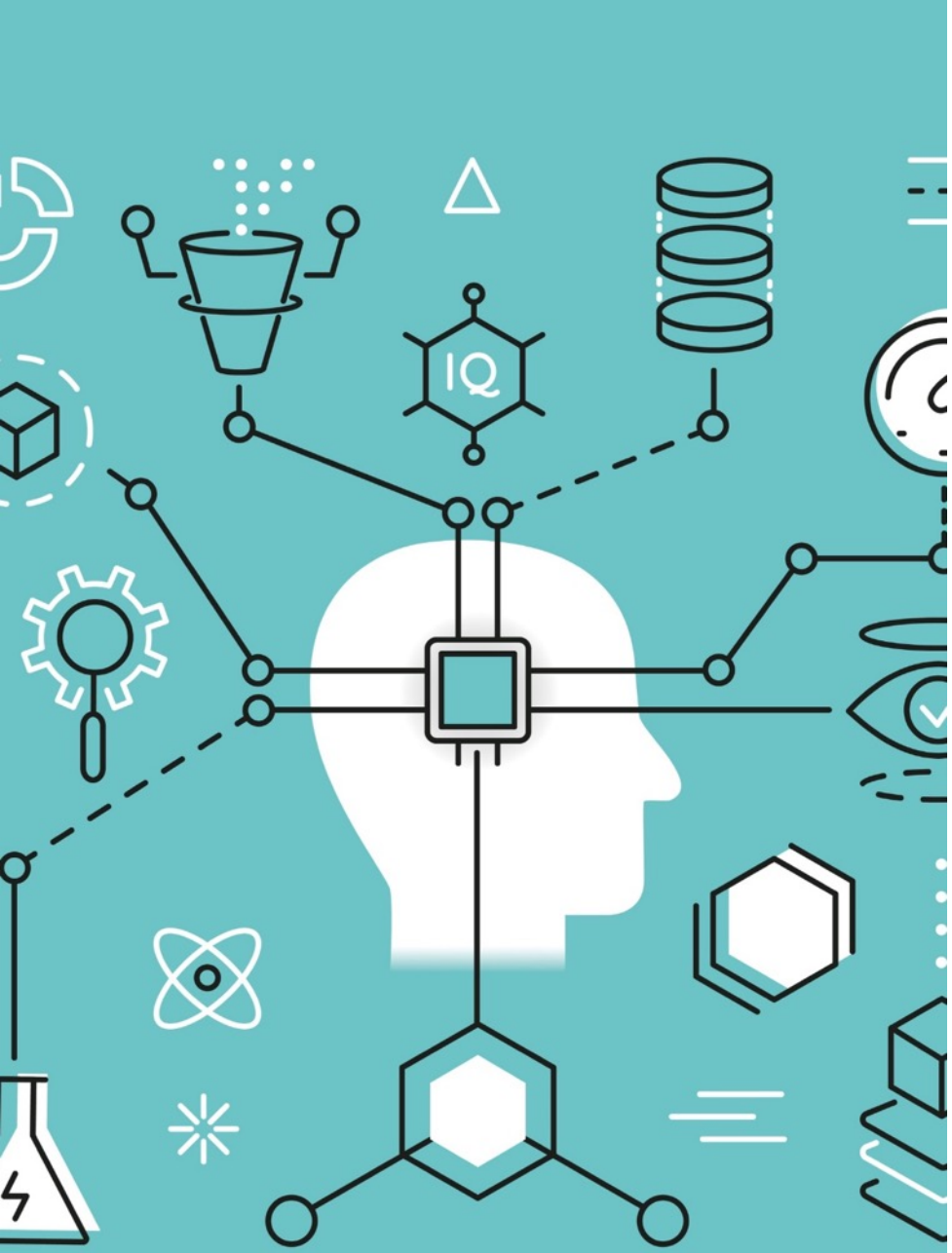
Split Training and Validation Sets

```
smp_size <- floor(0.75 * nrow(df))

set.seed(42)
train_ind <- sample(seq_len(nrow(df)), size = smp_size)

train <- df[train_ind, ]
test <- df[-train_ind, ]

model <- glm(Survived ~.,family=binomial(link='logit'),data=train)
```

R Example

Fit and Evaluate Model

```
pred <- predict(model,newdata=test,type='response')
pred_survived <- round(pred)

sprintf( "Accuracy: %f", sum(pred_survived == test$Survived) /
nrow(test) )
```



RGA

SAS

SAS (previously "Statistical Analysis System") is a commercial application/programming language introduced by SAS in 1976 by the Institute for Advanced Analytics. SAS is primarily targeted at statistical computing.

Reasons to Use SAS

Well-known statistical package with commercial support.

- It is easy to move between SPSS and SAS.
- Can use disk in addition to RAM – everything does not need to fit in RAM.
- Usually faster than R.
- Better debugging environment than R.
- Dedicated customer service support.
- Very stable and backwards compatible.

RGA

Reasons Not to Use SAS

Expensive and slow to adapt commercial package.

- Less community support – not as easy to “google an answer.”
- Licensing fees.
- Can take multiple years to add new statistical methods.
- One object only: the data set – no vectors, lists, or other objects.

https://science.nature.nps.gov/im/datamgmt/statistics/r/documents/r_for_sas_spss_users.pdf

RGA

SAS Example

Load Data and Preprocess

```
/* Read the CSV */
PROC IMPORT DBMS=csv OUT=train  REPLACE
  DATAFILE="/folders/myfolders/titanic-dataset.csv";
  GETNAMES=YES;
RUN;

/* Fill in missing ages with median */
PROC STDIZE DATA=train OUT=train
            METHOD=median reponly;
    VAR Age;
RUN;
```

SAS Example

Split Training and Validation

```
PROC SURVEYSELECT DATA=train outall OUT=train METHOD=srs
SAMPRATE=0.7;
RUN;
DATA validate;
      SET train;
      IF selected = 0;
RUN;
DATA train;
      SET train;
      IF selected = 1;
RUN;
```

SAS Example

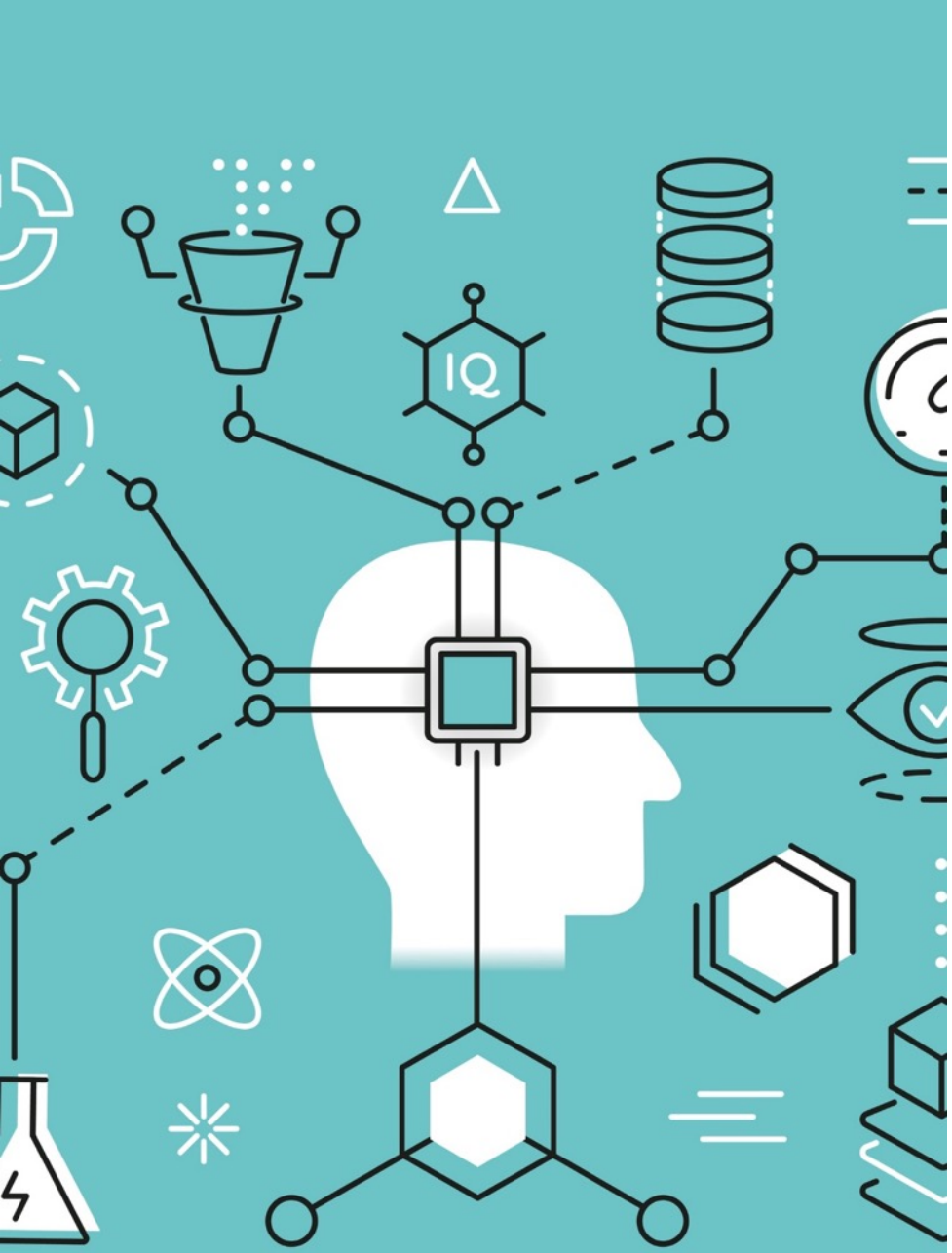
Fit Model and Predict

```
/* Fit the logit */
PROC LOGISTIC data=train outmodel=model descending;
  CLASS Sex / PARAM=ref ;
  CLASS Embarked / PARAM=ref ;
  MODEL Survived = Sex Age Pclass Parch SibSp Embarked;
RUN;

/* Predict */
PROC LOGISTIC INMODEL=model;
      SCORE DATA=validate OUT=pred;
RUN;
```

SAS Example

Evaluate

```
/* Turn prediction probabilities into class values (threshold=.5) */
DATA pred;
    SET PRED(KEEP = PassengerId Survived P_1);
    pred_survived = ROUND(P_1);
RUN;

/* Evaluate */
proc freq data=pred;
      tables Survived * pred_survived;
run;
```



RGA

PySpark

Though not a programming language itself, the Spark Python API (PySpark) exposes the Spark programming model to Python. This allows Python access to the big data and machine learning capabilities of Spark.

Reasons to Use PySpark

Bring the power of Spark into Python.

- Python code can be executed in parallel over large grids of computers.
- Extensive support for machine learning.
- Extensive support for data processing.
- Highly scalable through Spark.
- Performance issues of Python becomes less important on large grids.
- Compatible with the Java (JVM) ecosystem.

RGA

Reasons Not to Use PySpark

Might be overkill for smaller projects.

- Spark might be overkill, and slower, for smaller projects.
- Well known libraries, such as Scikit-Learn and Pandas are not directly compatible with PySpark.
- Libraries such as Dask may give the best of both worlds for small and large projects.

https://science.nature.nps.gov/im/datamgmt/statistics/r/documents/r_for_sas_spss_users.pdf

RGA

PySpark Example

Establish Session and Load Data

```
spark = SparkSession \
      .builder \
      .appName("Spark ML example on titanic data ") \
      .getOrCreate()

s3_bucket_path = "/mnt/lp-dataset/titanic/train.csv”

titanic_df = spark.read.csv(s3_bucket_path,header =
'True',inferSchema='True’)
passengers_count = titanic_df.count()
print(passengers_count)
```

PySpark Example

Preprocess Data

```
mean_age = titanic_df.select(mean('Age')).collect()[0][0]

titanic_df = titanic_df.na.fill({" 'Age' : mean_age })

titanic_df = titanic_df.na.fill({"Embarked" : 'S'})

titanic_df =
titanic_df.drop("PassengerId","Name","Ticket","Cabin","Embarked","Se
x","Initial", "Cabin")
```

PySpark Example

Generate Feature Vector and Split Train/Test

```
feature =
VectorAssembler(inputCols=titanic_df.columns[1:],outputCol="features
") feature_vector= feature.transform(titanic_df)

(trainingData, testData) = feature_vector.randomSplit([0.8,
0.2],seed = 11)
```

PySpark Example

Fit a Random Forest

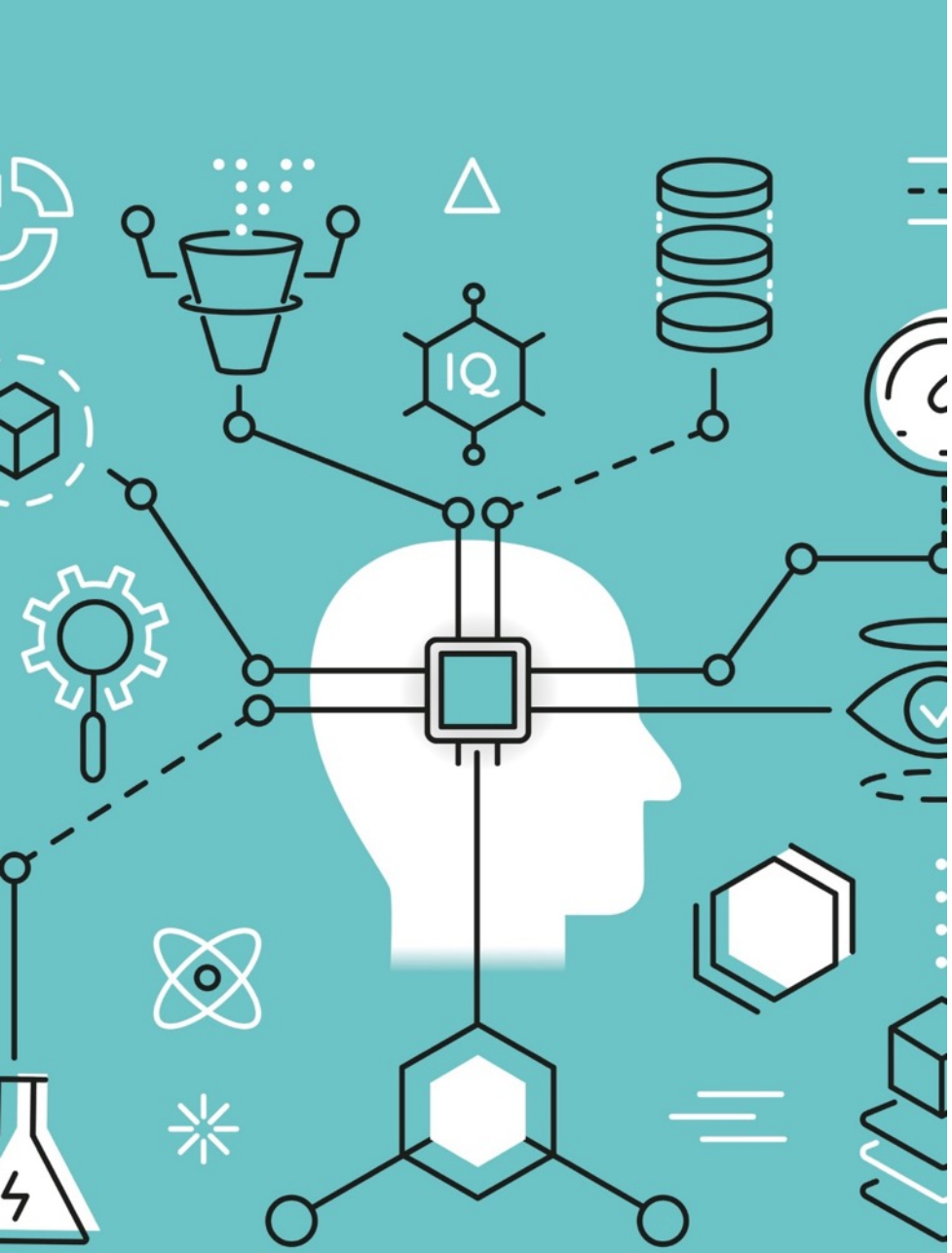
```
from pyspark.ml.classification import RandomForestClassifier

rf = DecisionTreeClassifier(labelCol="Survived",
featuresCol="features")
rf_model = rf.fit(trainingData)
rf_prediction = rf_model.transform(testData)
rf_prediction.select("prediction", "Survived", "features").show()
```

RGA

PySpark Example

Display Accuracy

```
rf_accuracy = evaluator.evaluate(rf_prediction)

print("Accuracy of RandomForestClassifier is = %g” % (rf_accuracy))

print("Test Error of RandomForestClassifier = %g " % (1.0 -
rf_accuracy))
```



RGA

Scala

Scala is a general-purpose programming language, created in 2004, that provides support for functional programming and a strong static type system. Spark was written in Scala and therefore has the most direct support of Spark possible.

Reasons to Use Scala

Functional, concurrent language with direct Spark support.

- Scala is the language that Spark was written in. There is no better Spark integration.
- Considerably faster than Python, even without the use of special libraries.
- Very advanced support for threading and true concurrency.
- Some prefer Scalas stronger type safety model.
- Machine learning models can be implemented completely in Scala, without the need for low-level languages like C++.

RGA

Reasons Not to Use Scala

Complex language with a difficult learning curve.

- Unless you have worked
- Licensing fees.
- Can take multiple years to add new statistical methods.
- One object only: the data set – no vectors, lists, or other objects.

https://science.nature.nps.gov/im/datamgmt/statistics/r/documents/r_for_sas_spss_users.pdf

RGA

Scala Example

Load Data

```
val trainDF = sqlContext.table("titanic_train")
      .withColumn("Age", $"Age".cast("double"))
      .withColumn("Pclass", $"Pclass".cast("int"))
```

Scala Example

Preprocess Data

```
val meanAge = trainDF.agg(avg("Age").as("Age_avg"))
      .collect()(0).getDouble(0)

val trainDF_cleaned =
      trainDF.na.fill(meanAge, Seq("Age"))
```

Scala Example

Generate Feature Vector

```
val labelIndexerModel = new StringIndexer()
      .setInputCol("Survived")
      .setOutputCol("label")
      .fit(trainDF_cleaned)

val trainDF_withLabel = labelIndexerModel.transform(trainDF_cleaned)

val label_indexToString = new IndexToString()
      .setInputCol("prediction")
      .setLabels(labelIndexerModel.labels)
      .setOutputCol("Survived_prediction")
```

RGA

Scala Example

Build ML Pipeline

```
val sexIndexerModel = new StringIndexer()
      .setInputCol("Sex")
      .setOutputCol("Sex_indexed")
      .fit(trainDF_withLabel)
val vectorAssembler = new VectorAssembler()
      .setInputCols(Array("Age", "Sex_indexed", "Pclass"))
      .setOutputCol("features")
val classifier = new DecisionTreeClassifier()
      .setMaxBins(32) .setMaxDepth(2)
      .setImpurity("gini").setSeed(42)
val pipeline = new Pipeline()
      .setStages(Array(sexIndexerModel, vectorAssembler,
      classifier, label_indexToString))
```

RGA

Scala Example

Fit a Model and Predict

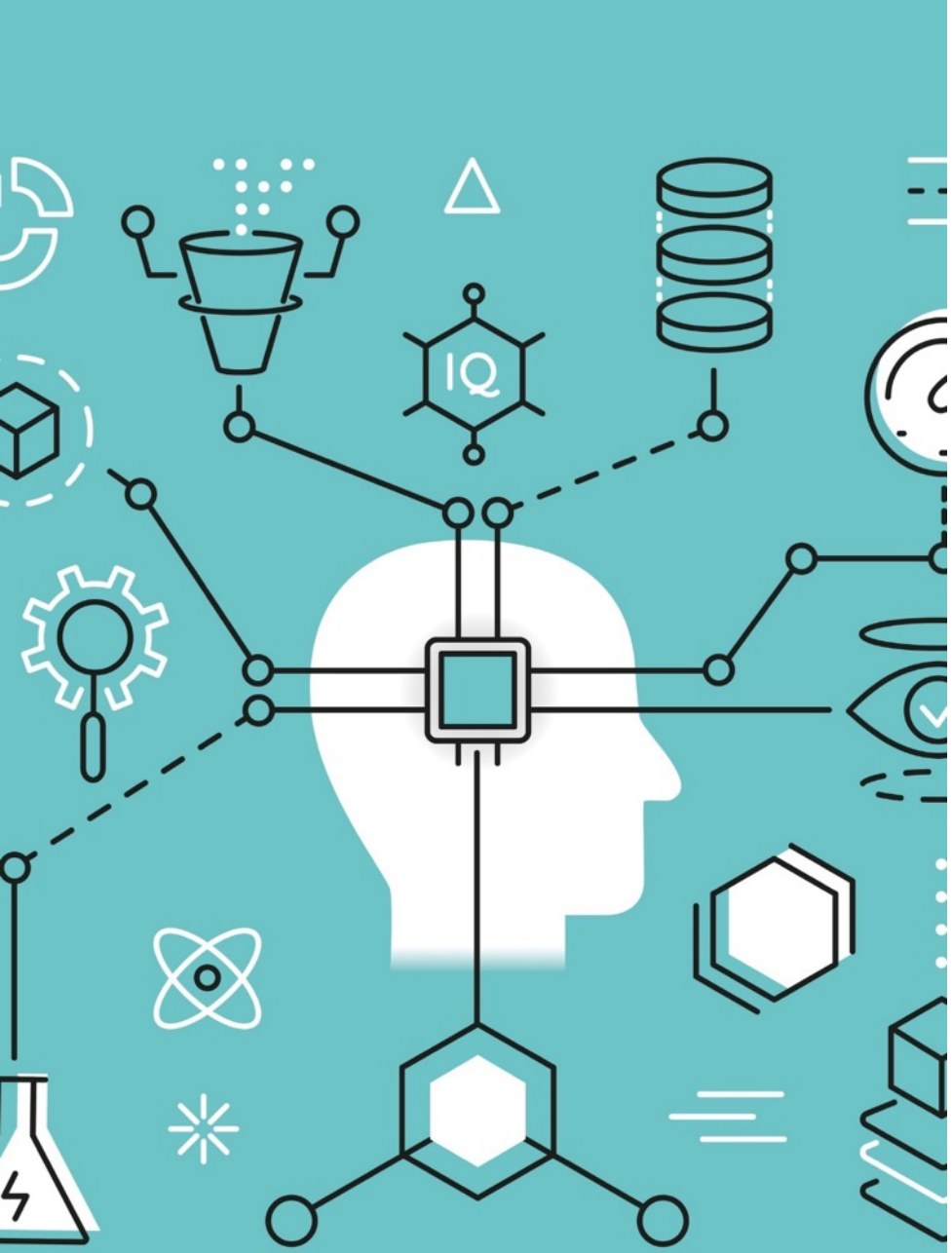
```
val pipelineModel = pipeline.fit(trainDF_withLabel)

val testDF_results = pipelineModel.transform(testDF)

val expectedResultsDF = spark.table("titanic_gender_submission")
val comparisonDF = testDF_results.select("PassengerId",
      "Survived_prediction") .join(expectedResultsDF,
      testDF_results("PassengerId") ===
            expectedResultsDF("PassengerId"))
      .drop(expectedResultsDF("PassengerId"))
```

RGA

Scala Example

Display Accuracy

```
val comparisonStatsDF = comparisonDF.groupBy("Survived",
"Survived_prediction").count()

display(comparisonStatsDF)
```



RGA

Thank you!

Any questions?


RGA