Figure 1: alt text

Introduction to R

author: Jan Vandepitte date: 30 August 2018 autosize: true —

0. What is R

- Open source Script language for statistical programming
- data preparation, machine learning, experimentation, visualization (presentation, notebook, shiny dashboard)
- packaging system (CRAN) like nuget, npm

```
x <- 1
x + 1
```

0.1 Origins in LISP

LISP -> Scheme -> S -> R (cfr Ecmascript) * REPL * lambda's * reflection (code is data) * dynamic (if it quacks like a duck) * abstract away underlying system (for domain experts) * backed by fast C++, C or Fortran libraries * column oriented data structure (APL influence)

http://paulgraham.com/icad.html

0.2 Origins in LISP (demo)

```
myF <- function(x) { x+1 }
myF
myF(1)
apply(matrix(c(1,2,3,4),2,2),1,myF)
```

Figure 2: alt text

1. Why R

- AI coming out of winter (pit of dispair)
- Omnipresent in AI (Skill like SQL, Ecmascript)
- Open source, Eco-system, History (de facto standard)
- different backends (citizen scientist, Big Data, Keras backend)
- Backed by and integrated in Microsoft products

1.1 Notable R tools in ecosystem

- packages: https://cran.r-project.org/
- IDE Rstudio: https://www.rstudio.com/
- notable R packages:
- dplyr and derivatives: high level data wrangling
- shiny : Interactive dashboarding
- RCurl : Get data from an API
- knitR: Mix markdown and R (this pres)
- ggplot2 : Nice graphs
- lattice : Multivariate graphs
- devtools : Get packages straight from github and more fun
- tableplot: visualize big datasets

1.2 Checklist: Is R the tool for me right now?

- Do I want to use free (like beer) software : yes
- Do I want to experiment with different models from different creators : yes
- Does my data fit in main memory of my computer (no big data): yes
- Doesn't my data fit in main memory of my computer : yes with some prerequisites

2. R in Microsoft products

- 2015 Microsoft buys Revolution Analytics
- R Integrated in several products
- https://mran.microsoft.com

2.1 Microsoft R Open (R extension)

Improved some pittfals of typical R distribution (single threaded, standardization of packages and models, object orientation. . .) CRAN

(see Typescript)

Figure 3: alt text

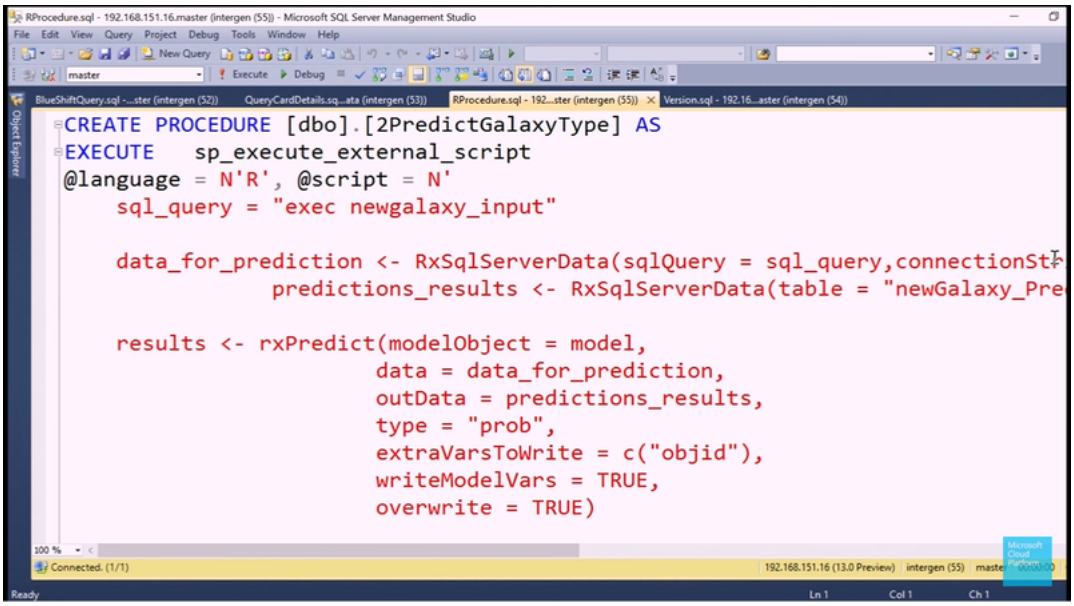

Figure 4: alt text

2.2 Microsoft Machine Learning Server (run time)

Client/Server Operationalizing R * DeployR * ScaleR

Now also python

2.3 Microsoft R Archive Network (MRAN)

Standardized package, snapshotting in time https://mran.microsoft.com/

2.4 R in SQL Server

Machine learning services in SQL Server

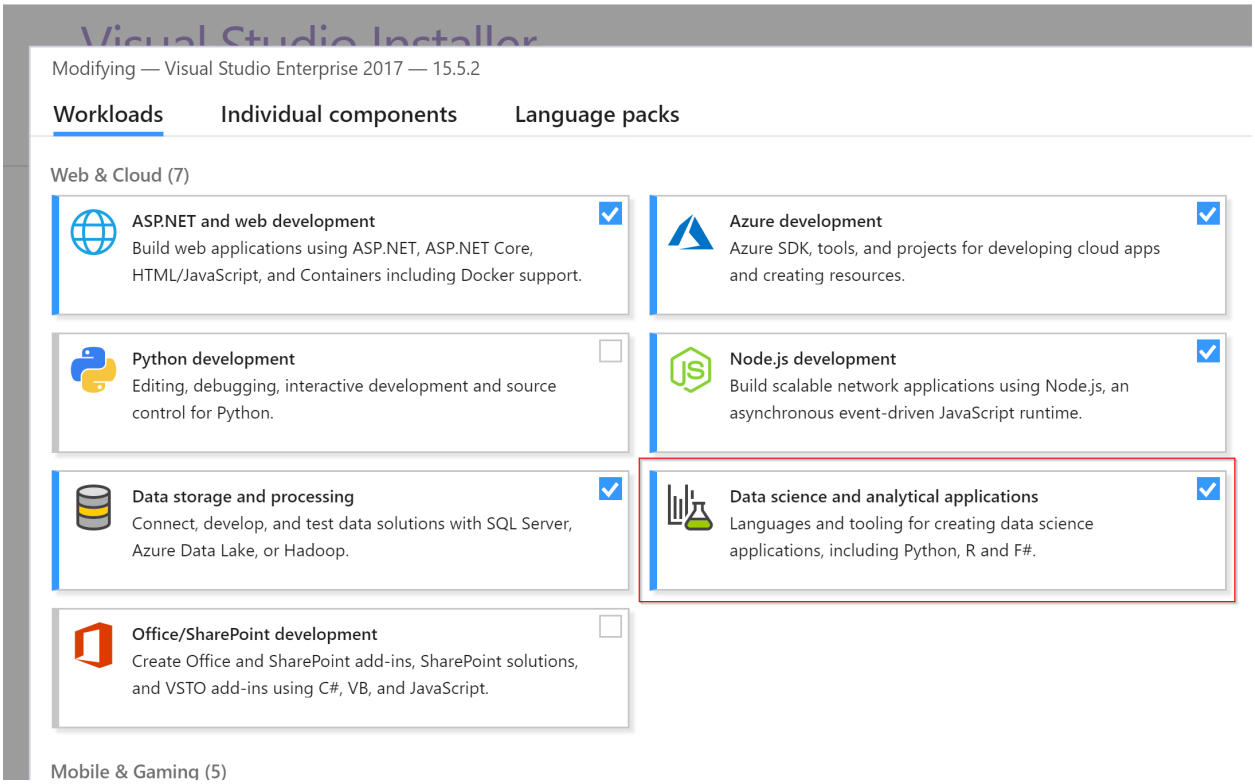

Figure 5: alt text

2.5 R in visual studio

2.6 R in Power BI

Visualization with R? PowerBIR?? (jk :p)

2.7 Azure machine learning

2.8 Azure HDInsight (hadoop as a service)

http://blog.revolutionanalytics.com/2015/06/using-hadoop-with-r-it-depends.html

2.9 Azure Databricks (spark as a service)

R one of the languages on Apache Spark (besides Scala and Python)

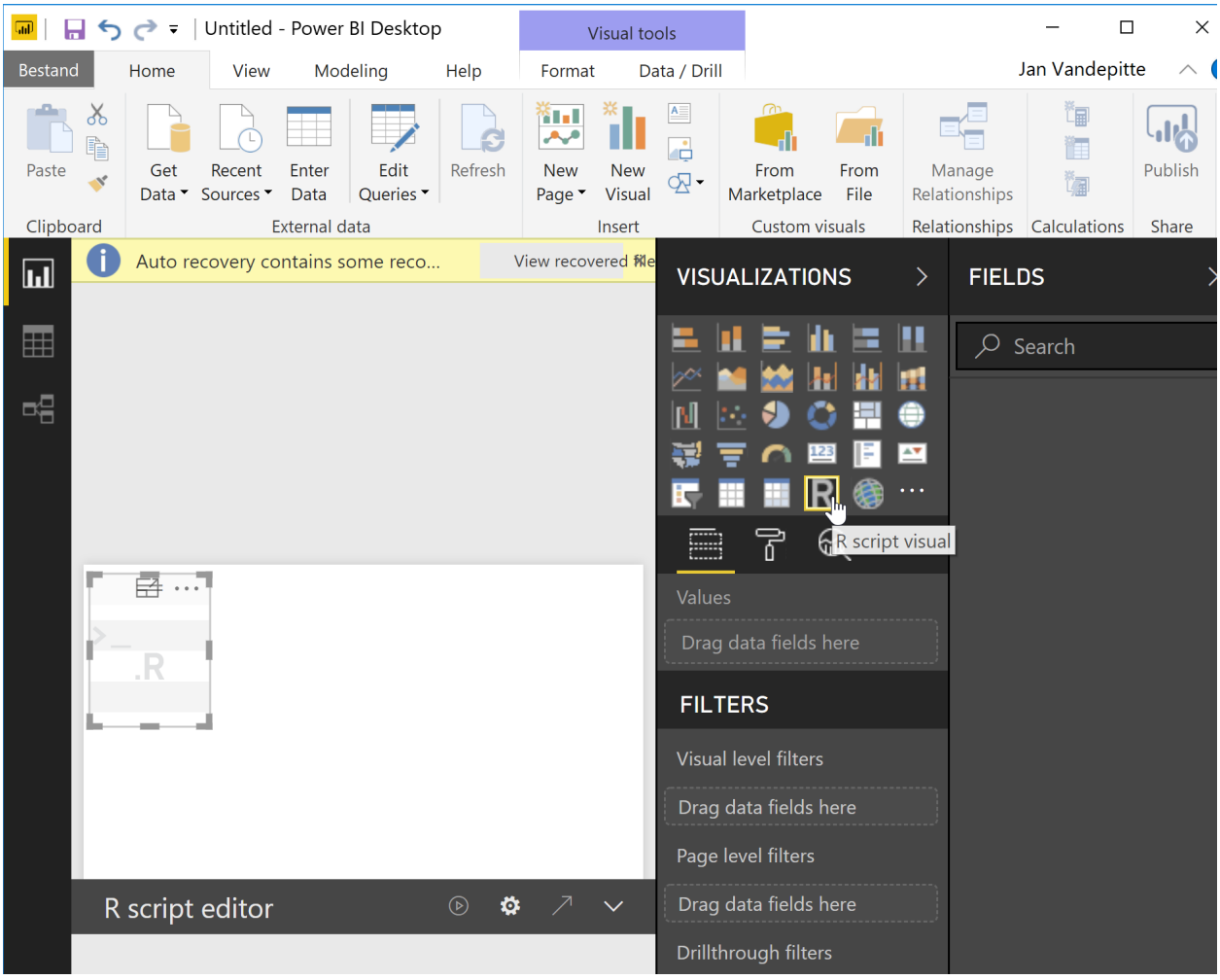

Figure 6: alt text

NAAM	UITGEVER	CATEGORIE
Machine Learning Studio Workspace	Microsoft	Machine Learning
Machine Learning Modelbeheer (preview)	Microsoft	Machine Learning
Machine Learning Experimentation (preview)	Microsoft	Machine Learning
Machine Learning Studio Web Service	Microsoft	Machine Learning
Machine Learning Studio Web Service Plan	Microsoft	Machine Learning
Deep Learning Virtual Machine	Microsoft	Machine Learning
Data Science Virtual Machine for Linux (Ubuntu)	Microsoft	Machine Learning
Data Science Virtual Machine for Linux (CentOS)	Microsoft	Machine Learning
Data Science Virtual Machine - Windows 2016	Microsoft	Machine Learning
Data Science Virtual Machine - Windows 2012	Microsoft	Machine Learning

Figure 7: alt text

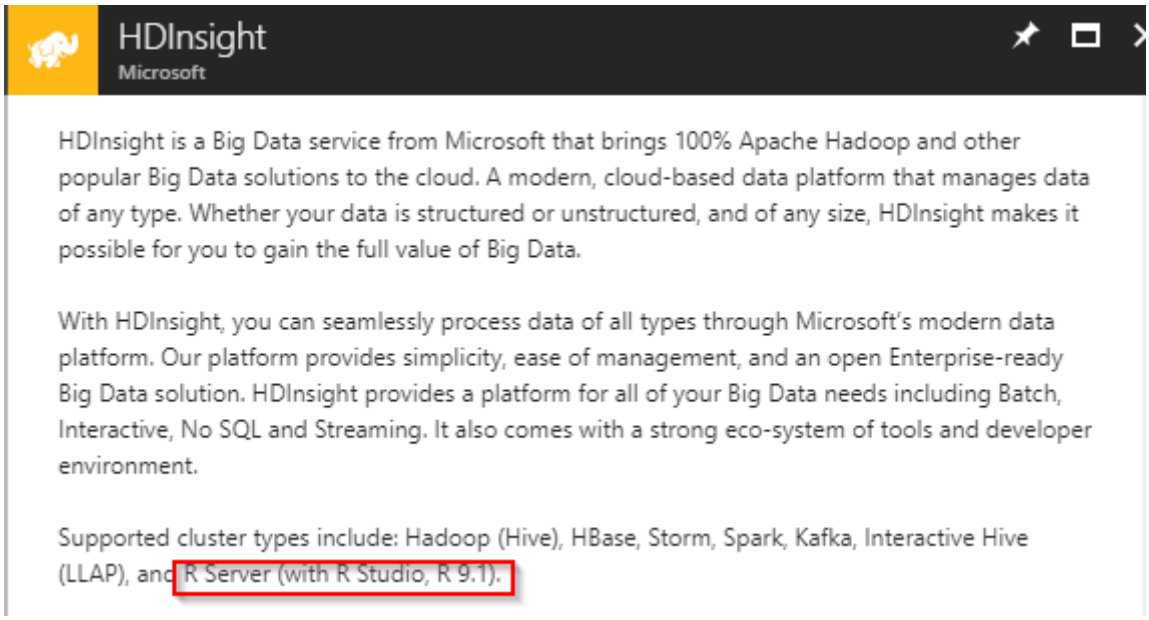

Figure 8: alt text

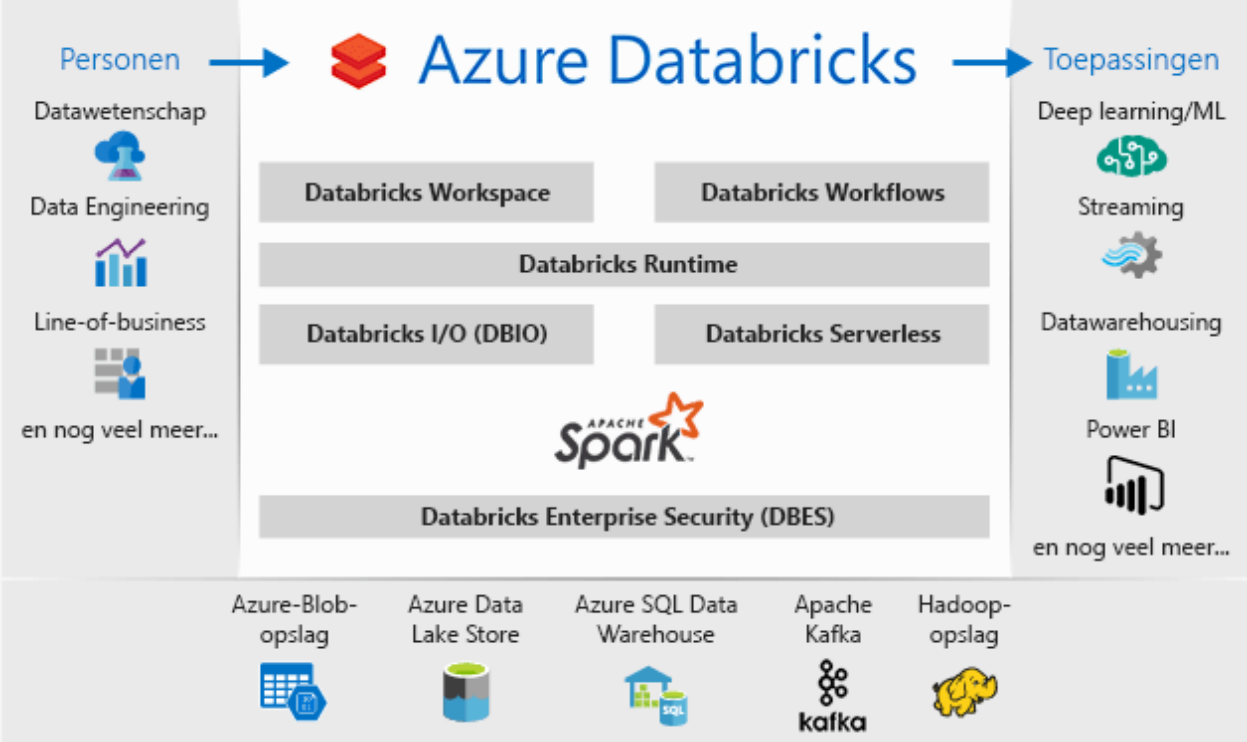

Figure 9: alt text

2.10 In minecraft

https://ropenscilabs.github.io/miner_book/

2.11 Certification as a Data Scientist

3. Concepts

In the following segment I try to provide a high level run through of basic concepts of AI, machine learning and statistics

3.1 AI

3.2 DIKW pyramid

```
{r, echo=FALSE, out.width = "4000px"} knitr::include_graphics("Introduction-to-R-figure/DIKWpyramid.gif
```

3.3 DIKW cycle

```
{r, echo=FALSE, out.width = "4000px"} knitr::include_graphics("Introduction-to-R-figure/DIKWcycle.jpg")
```

R Programming with Minecraft

Brooke Anderson, Karl Broman, Gergely Daróczi, Mario Inchiosa, David Smith, and Ali Zaidi

Figure 10: alt text

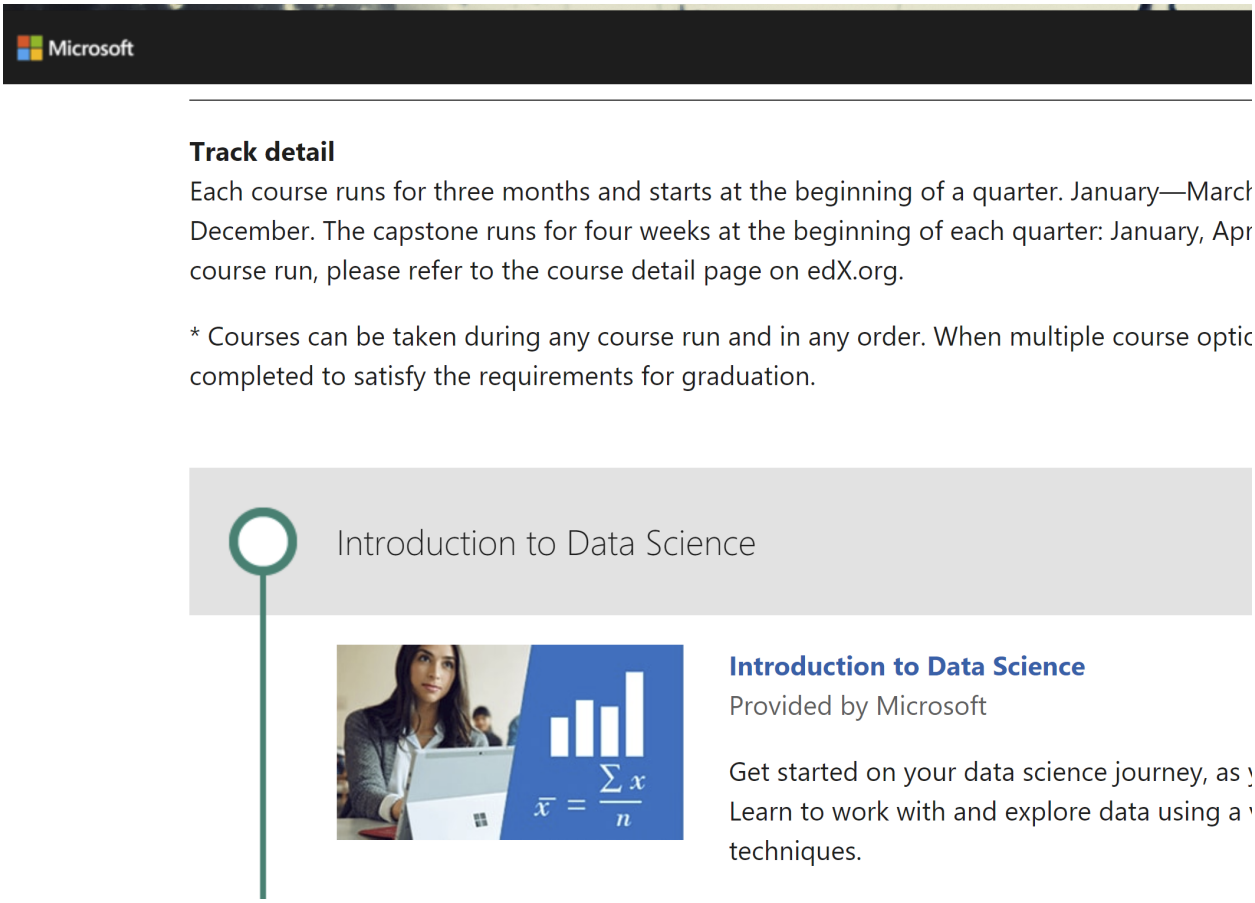

Figure 11: alt text

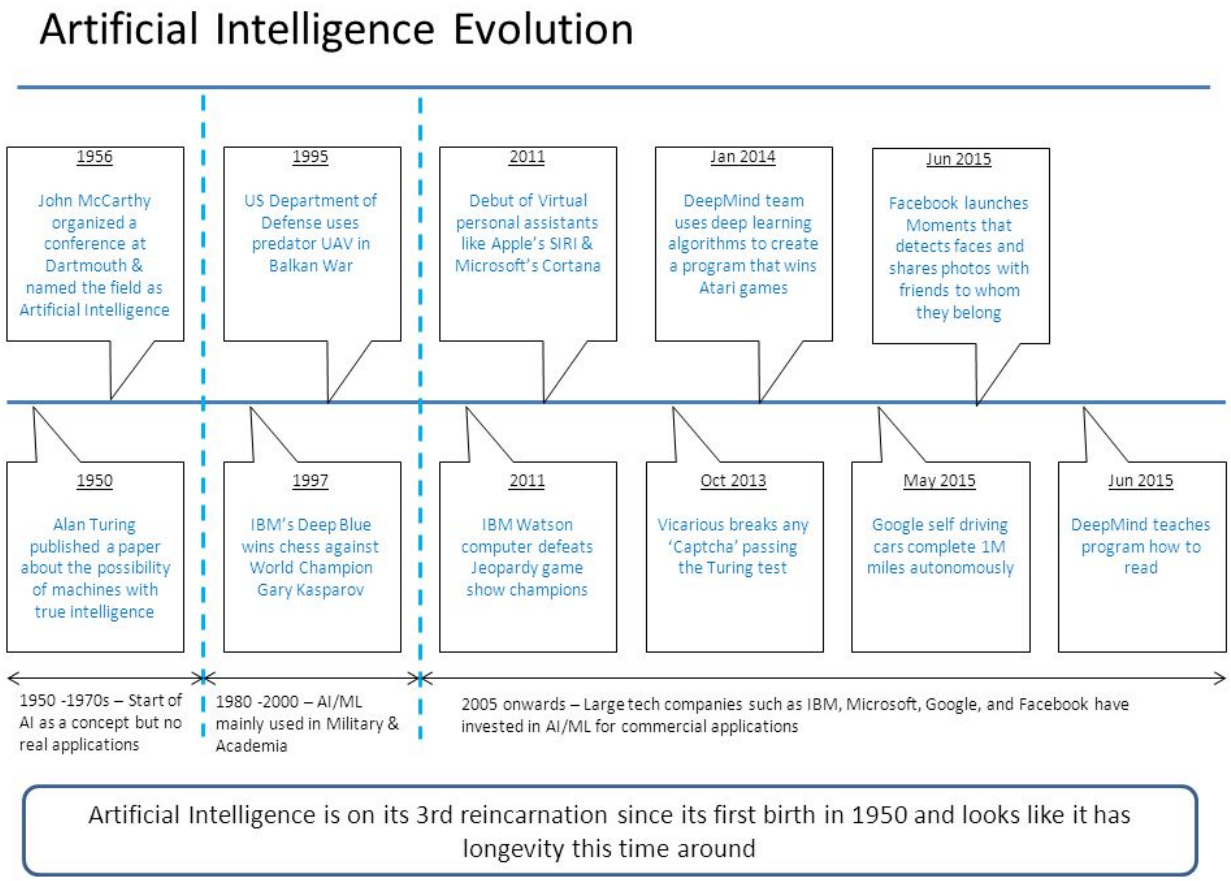

Figure 12: alt text

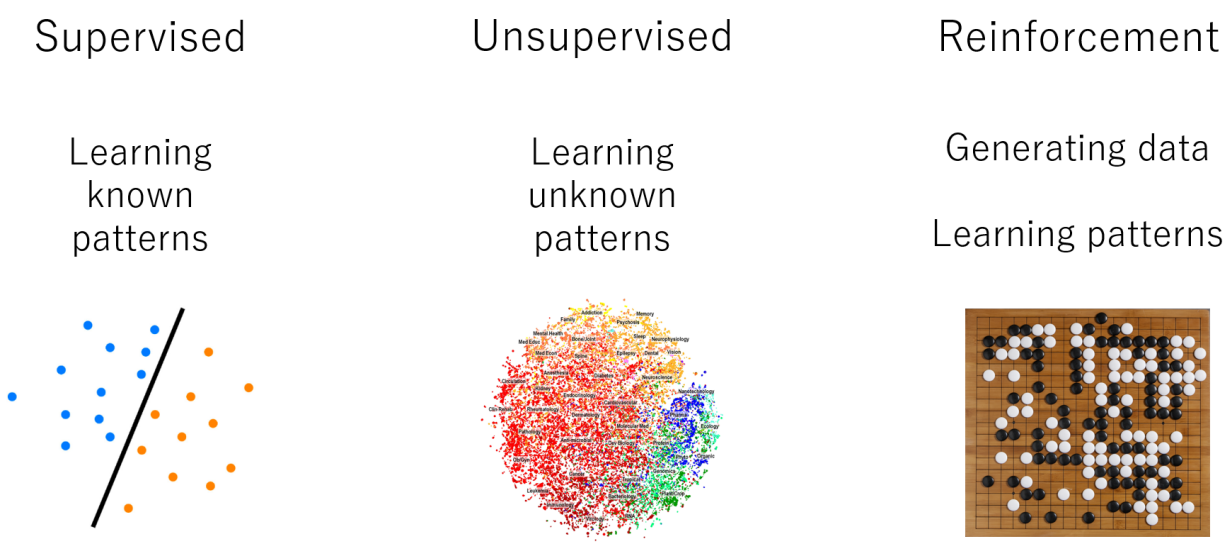

Figure 13: alt text

3.4 Statistics and Machine learning

3.5 Error terms

In programming we try to reduce the errors in our models (programs) by fixing bugs and doing unit testing.

Reduce Errors Of Our statistical models : Stochastical element. Error is quantifiable with data.

3.5 Error terms

```
{r, echo=FALSE, out.width = "4000px"} knitr::include_graphics("Introduction-to-R-figure/residual.png")
```

Model is estimation of errorterms of data around model

3.5 Error terms

```
{r, echo=FALSE, out.width = "4000px"} knitr::include_graphics("Introduction-to-R-figure/errorterm2.jpg"
```

Error has a probability distribution around: assumed to be normal distributed

3.6 Error types : hypothesis testing

```
{r, echo=FALSE, out.width = "4000px"} knitr::include_graphics("Introduction-to-R-figure/errortypes.png"
```

3.6 Error types : hypothesis testing

- type I : incorrectly detect effect when there is none (bias, noise...) : overfitting

- type II : incorrectly detect no effect (0 hypothesis) when there is an effect : underfitting

Programming = mathematical proof testen (https://en.wikipedia.org/wiki/Curry%E2%80%93Howard_correspondence)

3.7 Information = Variance

we train our model and test/validate it on the same data

3.7 Information = Variance

How related is our training, test our validation data subset? Idempotency of our model.

3.8 Big data - Central Limit Theorem

Larger sample size: more normal distribution

with more data, we need less assumptions of underlying error term

3.9 Big data - Power increases with sample size

Power is chance that our model will correctly detect an effect. Power will decrease with more features in our model. Power will increase with larger sample size for our model. More power = less underfitting (less type II error).

Median vs arithmetic average.

3.10 Big data - Data is the new oil

Conclusion: the more (quality) data, the better for our models

New headaches: Horizontal scalability, distributed systems, CAP and CALM theoreum ...

3.11 Deep learning

deep neural networks, long history, more data and faster hardware (GPU and even TPU)

> When in doubt use brute force ~ Ken Thompson

3.12 Deep learning

Renewed interest so some interesting developments * Auto-encoder (encoder-decoder chained learned with internal representation) * SEQ2SEQ (arbitrary size representation for e.g. NLP) * Representation learning (less feature engineering) * Auto-ML : deep learning for non-experts * cross platform: tensorflow on RPI and in browser * Theoretical framework (category theory): a new way of programming but with linear algebra http://colah.github.io/posts/2015-09-NN-Types-FP/

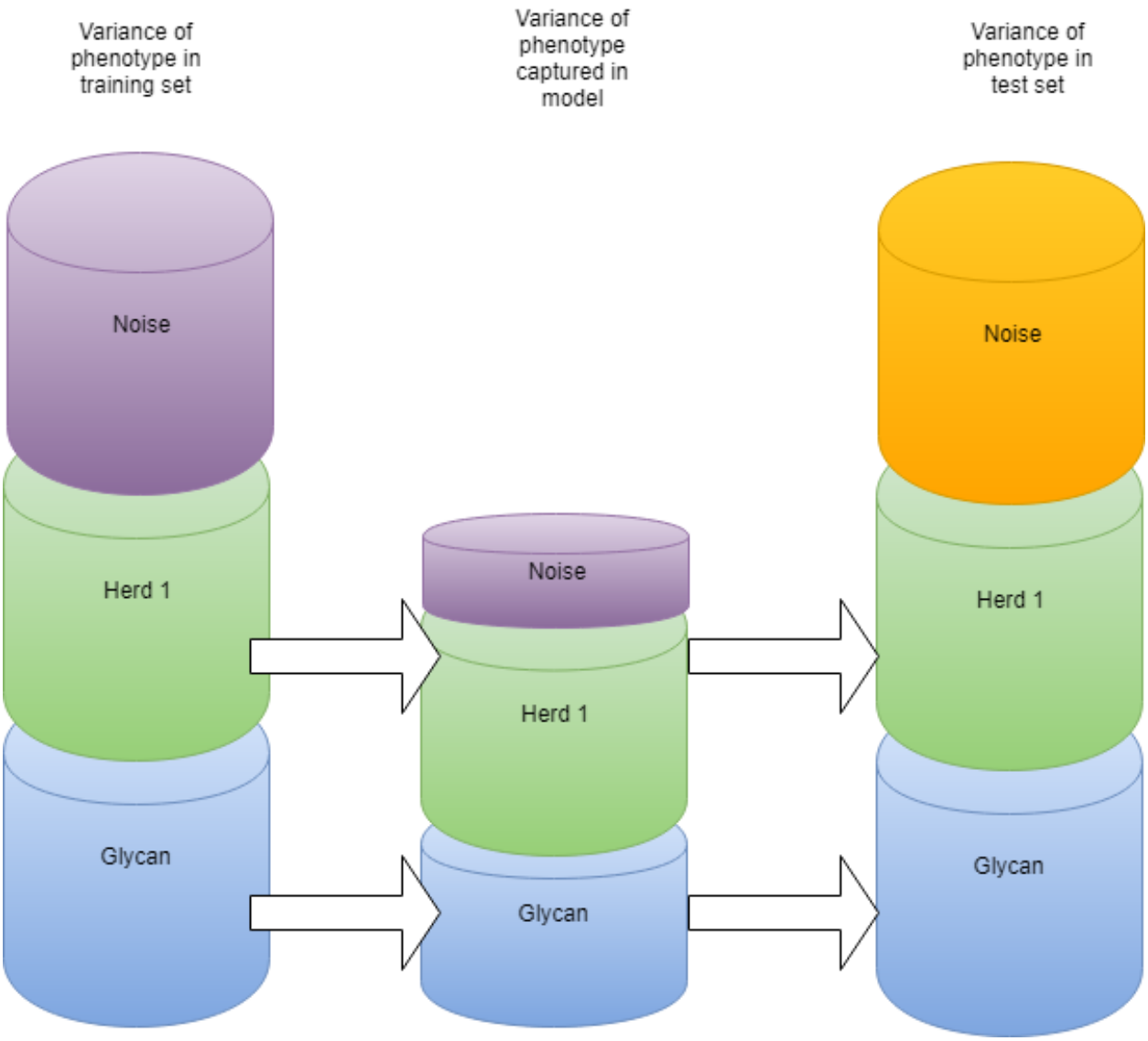

Figure 14: alt text

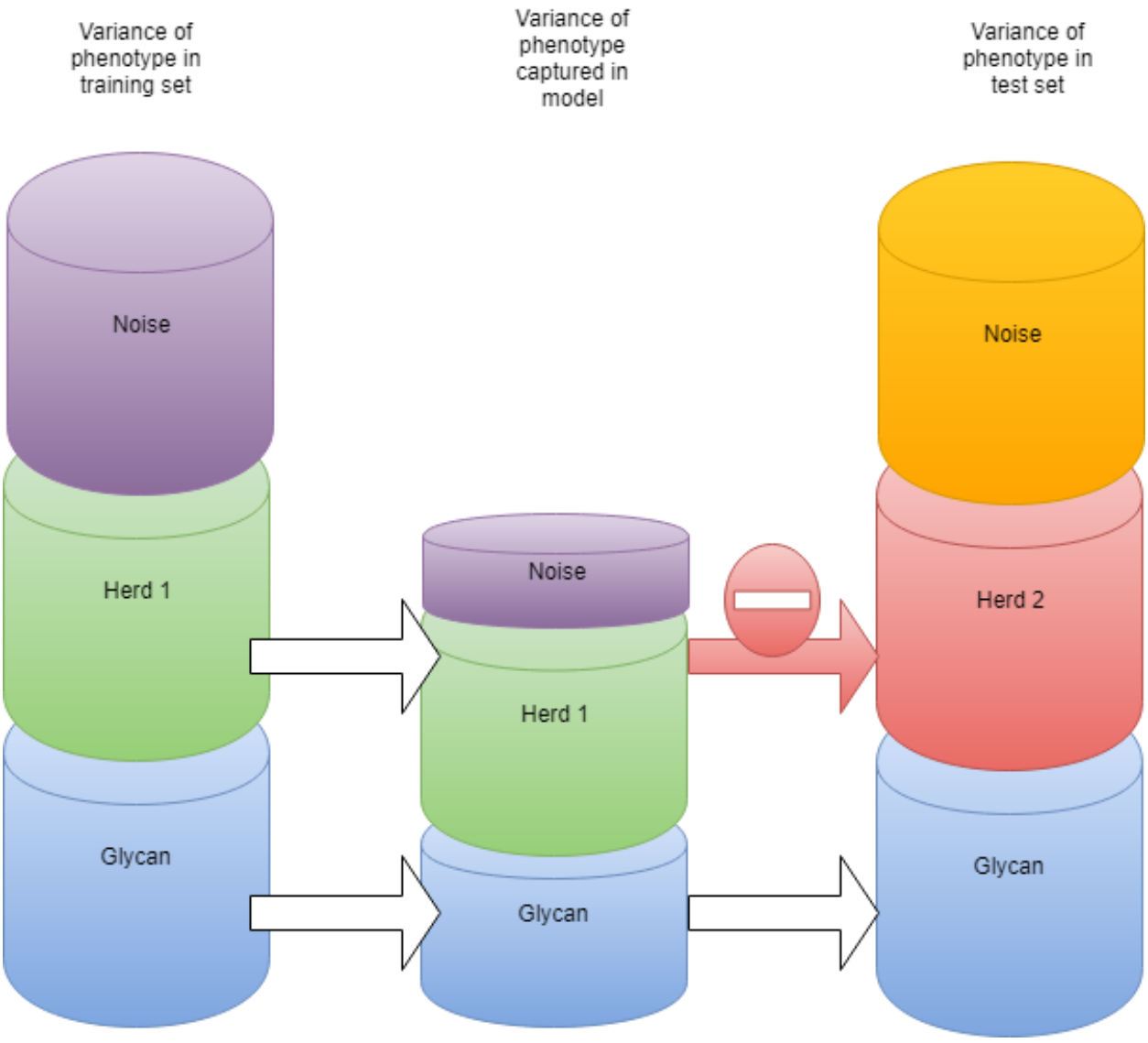

Figure 15: alt text

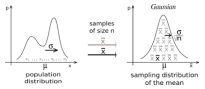

Figure 16: alt text

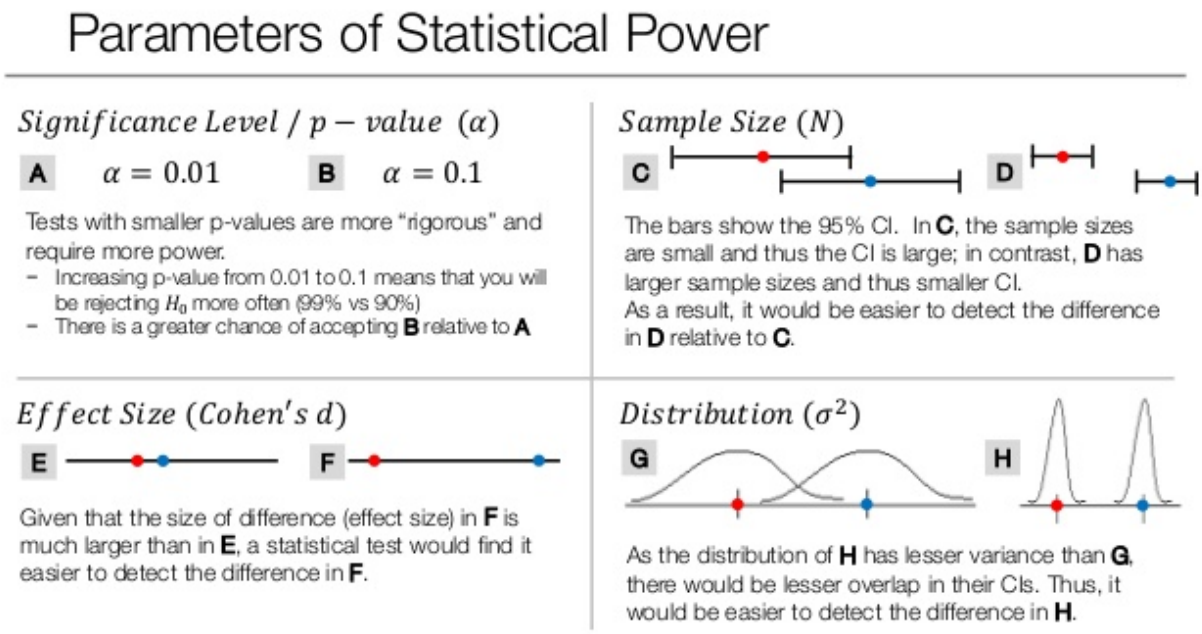

Figure 17: alt text

Figure 18: alt text

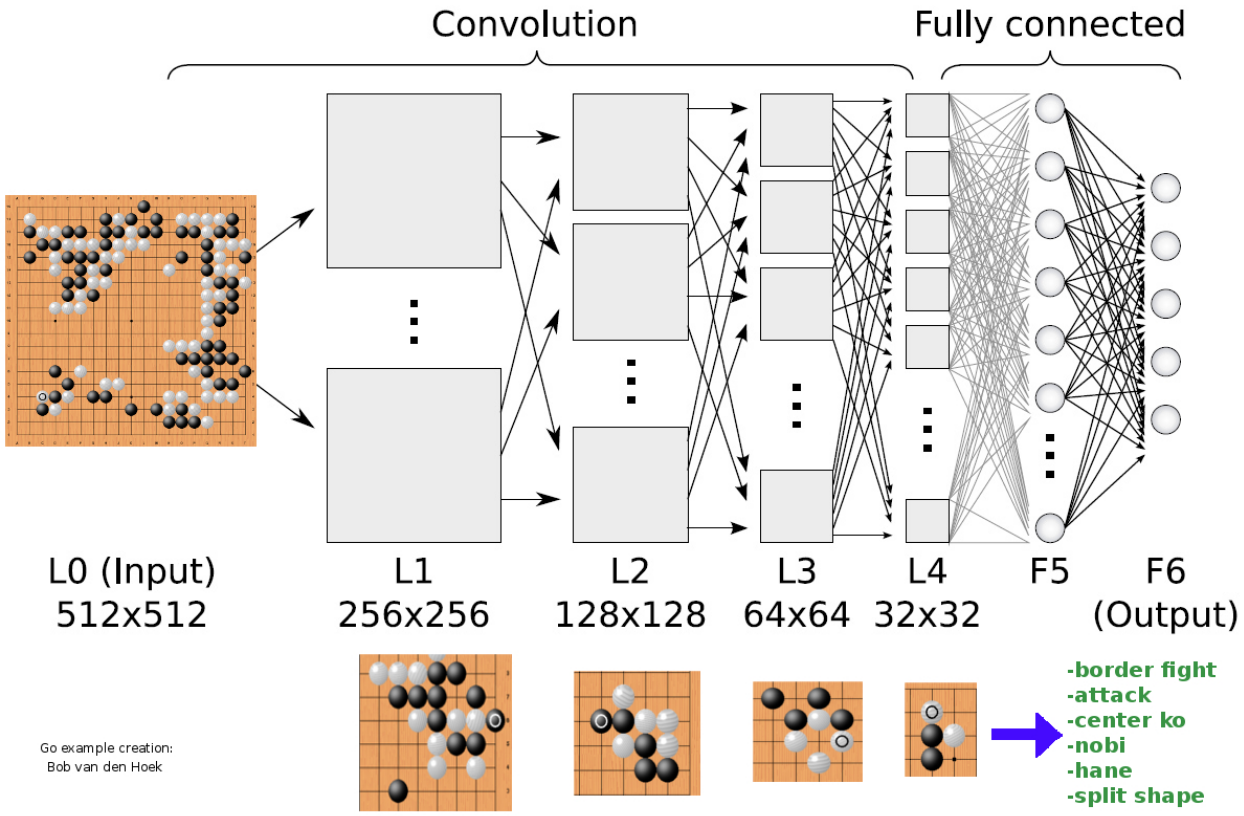

Figure 19: alt text

The Three Layer Causal Hierarchy

Level (Symbol)	Typical Activity	Typical Questions	Examples
1. Association $P(y\|x)$	Seeing	What is? How would seeing X change my belief in$Y$?	What does a symptom tell me about a disease? What does a survey tell us about the election results?
2. Intervention $P(y\|do(x), z)$	Doing Intervening	What if? What if I do X?	What if I take aspirin, will my headache be cured? What if we ban cigarettes?
3. Counterfactuals $P(y_x\|x', y')$	Imagining, Retrospection	Why? Was it X that caused Y? What if I had acted differently?	Was it the aspirin that stopped my headache? Would Kennedy be alive had Oswald not shot him? What if I had not been smoking the past 2 years?

Figure 1: The Causal Hierarchy. Questions at level i can only be answered if information from level i or higher is available.

Figure 20: alt text

3.13 Deep learning

Deep learning est mort, vive differentiable programming ~ Yann LeCun - Chief AI facebook

https://medium.com/@karpathy/software-2-0-a64152b37c35 https://www.facebook.com/yann.lecun/posts/10155003011462143 https://techburst.io/deep-learning-est-mort-vive-differentiable-programming-5060d3c55074

3.14 Deep dive theoretical

- Basic statistics : https://www.itl.nist.gov/div898/handbook/
- Machine learning : https://dzone.com/articles/35-free-online-books-machine
- https://www.kdnuggets.com/
- https://www.kaggle.com/
- https://www.datasciencecentral.com/

3.15 Level I in Causal hierarchy

https://arxiv.org/pdf/1801.04016.pdf

https://www.quantamagazine.org/to-build-truly-intelligent-machines-teach-them-cause-and-effect-20180515/

3.16 Simpson paradox

body weight related to disease (men vs women, children vs adults)

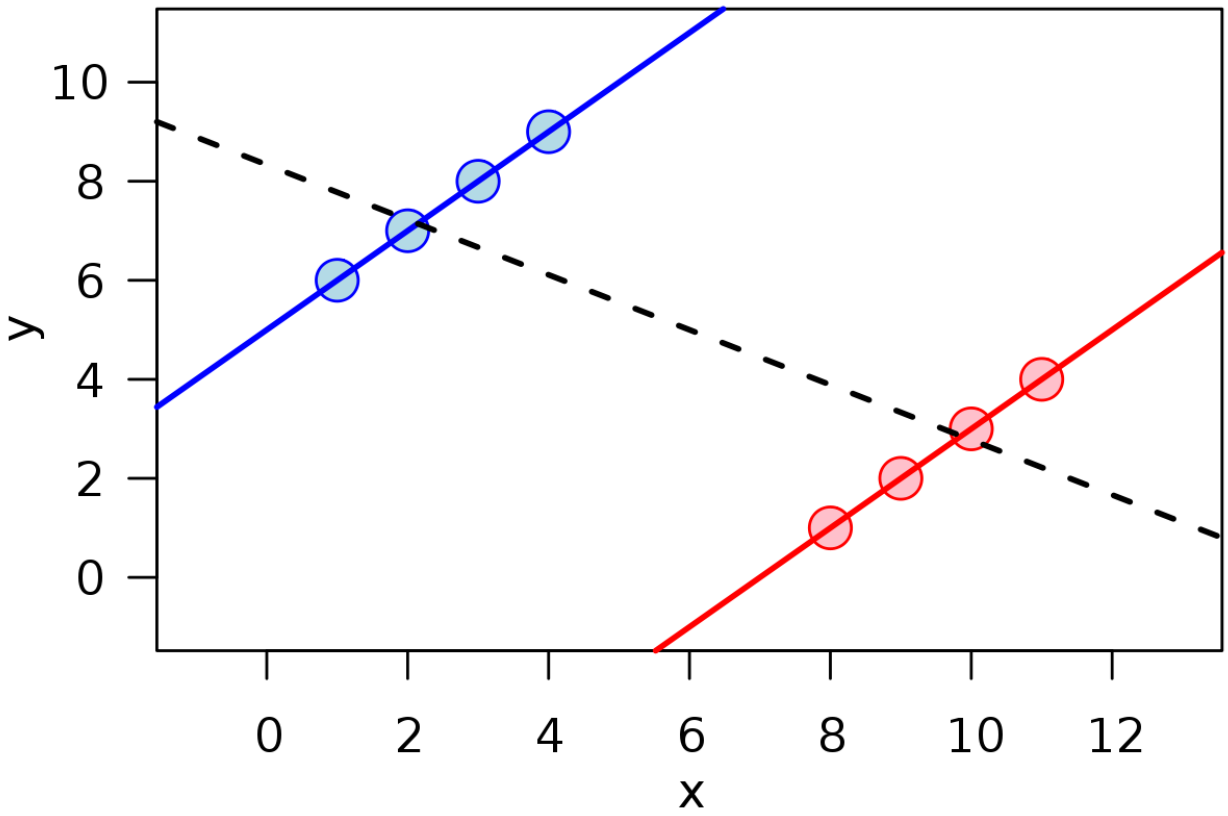

Figure 21: alt text

4. R for the .NET programmer

In knitR it's also possible to run code from other languages: * https://yihui.name/knitr/demo/engines/ * http://datadrivensecurity.info/blog/posts/2015/Jun/running-other-languages-in-r-markdown-files/

So let's add an engine for .net `{r setup, eval=FALSE} eng_dotnet <- function(options) { # create a temporary file f <- basename(tempfile("dotnet", '.', paste('.', "dotnet", sep = ''))) on.exit(unlink(f)) # cleanup temp file on function exit writeLines(options$code, f) out <- '' # if eval != FALSE compile/run the code, preserving output if (options$eval) { out <- system(sprintf('dotnet script %s', paste(f, options$engine.opts)), intern=TRUE) } # spit back stuff to the user engine_output(options, options$code, out) } knitr::knit_engines$set` 4.0 Let's try out our R-package ==========

```
if (!requireNamespace("devtools", quietly = TRUE)) { install.packages("devtools") }
devtools::install_github("yenwel/Rpresdotnetengine", force=T)
library(Rpresdotnetengine)
knitr::knit_engines$set(dotnet=eng_dotnet)
```

4.1 Let's try out our new engine

```
{r dotnet-ex, engine='dotnet', eval=TRUE, echo=TRUE} var i = 1; i++; Console.WriteLine(i++);
Console.WriteLine(++i); Console.WriteLine(i--); Console.WriteLine(--i);
```

4.2 Basic types (.NET)

```
{r dotnet-ex2, engine='dotnet', eval=TRUE, echo=TRUE} Console.WriteLine(1); Console.WriteLine(true);
Console.WriteLine("hello world?"); enum Color {Red, Green, Blue};  Console.WriteLine(Color.Red);
```

4.2 Basic types (R)

```
1
2.0
T
'hello world?'
as.factor(c('Red','Green','Blue'))[1];
as.ordered(c('Best','Bester','Bestest'))[1];
```

4.3 Collections and composite types (.NET)

```
{r dotnet-ex3, engine='dotnet', eval=TRUE, echo=TRUE}  Console.WriteLine(new [] { 1 ,
2 , 3}[1]); Console.WriteLine(new List<object> { "Fred" , 20}[0]); Console.WriteLine(new
Dictionary<string,object>{{"name","Fred"},{"age",20}}["name"]); Console.WriteLine(new {
Name = "Fred", Age = 20});
```

4.3 Collections and composite types (R)

```
c(1,2,3)
```

```
matrix(1:9, nrow=3,ncol=3)
list(name="Fred",  age=20)
# R has at least three ways to do OO (S3, S4, Reference class) but don't bother do FP rather
```

4.3 Collections and composite types (R)

```
array(1:16,dim = c(2,2,2,2))[,,2,2]
data.frame(name = c("Buddy", "Lisa"), age = c(10, 38), sex = as.factor(c("m","f")))
```

4.4 Functions (.NET)

```
{r dotnet-ex4, engine='dotnet', eval=TRUE, echo=TRUE} Func<int,int> myF = (int x) => x +
1; Console.WriteLine(myF); Console.WriteLine(myF(1)); Console.WriteLine(new [] {1 , 2 , 3
, 4}.Select(myF).FirstOrDefault());
```

4.4 Functions (R)

```
myF <- function(x) { x+1 }
myF
myF(1)
apply(matrix(c(1,2,3,4),2,2),1,myF)
```

4.5 Deep dive into R

start here: * https://www.statmethods.net/ * https://www.r-bloggers.com/ * https://www.datacamp.com/

then google (CRAN because R is to confusing for google)

```
{r, eval=FALSE} ??something
```

5. Demo's

- supply chain analysis :
  - https://github.com/yenwel/SCOperationsInventory
  - https://github.com/yenwel/supplychainplanning
- shiny app connecting to database: https://github.com/yenwel/shinyDatabaseExplorer
- analyse load tests (connect to db): https://github.com/yenwel/analyse-neustar-loadtest
- process IIS Url Rewrite xml: https://github.com/yenwel/processsUrlRewrite
- this presentation: https://github.com/yenwel/R-presentation
- An R-package for the knitr dotnet engine: https://github.com/yenwel/Rpresdotnetengine

5.1 Shiny App

5.2 data mining bitcoin and twitter

```
{r, eval=FALSE} #http://beautifuldata.net/2015/01/querying-the-bitcoin-blockchain-with-r/
library(Rbitcoin) trades <- market.api.process('kraken',c('BTC','EUR'),'trades') Rbitcoin.plot(trades,
col='blue')   {r, echo=FALSE, eval=FALSE} save(trades,file="trades.Rda")   {r, echo=FALSE}
load("trades.Rda")
```

5.2 data mining bitcoin and twitter

```
{r, echo=FALSE} #https://www.earthdatascience.org/courses/earth-analytics/get-data-using-apis/use-twitte
## install devtools package if it's not already #if (!requireNamespace("devtools", quietly
= TRUE)) { install.packages("devtools") } ## install dev version of rtweet from github
#devtools::install_github("mkearney/rtweet")       {r, eval=FALSE} ## load rtweet package
library(rtweet) # these environment variables are set via an app you can make at https://developer.twitt
envvar <- Sys.getenv(c("TWT_R_APP", "TWT_R_API", "TWT_R_SECRET","TWT_R_XS_TOKEN","TWT_R_XS_SECRET"))
appname <- envvar[1] key <- envvar[2] secret <- envvar[3] access_token <- envvar[4]
access_secret <- envvar[5] ## authenticate via access token token <- create_token(   app
= appname,    consumer_key = key,    consumer_secret = secret,    access_token = access_token,
access_secret = access_secret) bitcoin_tweets <- search_tweets(q = "#bitcoin", n = 15000
, retryonratelimit=F)   {r, echo=FALSE, eval=FALSE} save(bitcoin_tweets,file="tweets.Rda")
{r, echo=FALSE} load("tweets.Rda")
```

5.2 datamining bitcoin and twitter

```
str(trades)
```

5.2 datamining bitcoin and twitter

```
summary(trades)
```

5.2 datamining bitcoin and twitter

```
summary(trades$trades)
```

5.2 datamining bitcoin and twitter

```
summary(trades$trades$date)
```

5.2 datamining bitcoin and twitter

```
str(bitcoin_tweets)
```

5.2 datamining bitcoin and twitter

```
summary(bitcoin_tweets)
```

5.2 datamining bitcoin and twitter

```
summary(bitcoin_tweets$created_at)
```

5.2 datamining bitcoin and twitter

```
mintime <-max(min(trades$trades$date),min(bitcoin_tweets$created_at))
maxtime <- min(max(bitcoin_tweets$created_at),max(trades$trades$date))
tradesinrange <- trades$trades[trades$trades$date >= mintime & trades$trades$date <= maxtime,]
tweetsinrange <- bitcoin_tweets[bitcoin_tweets$created_at >= mintime & bitcoin_tweets$created_at <= max
mintime
maxtime
summary(tradesinrange$date)
summary(tweetsinrange$created_at)
```

5.2 datamining bitcoin and twitter

```
{r, out.width = "4000px"} hist(tradesinrange$date, "secs")
```

5.2 datamining bitcoin and twitter

```
{r, out.width = "4000px"} hist(tweetsinrange$created_at, "secs")
```

```
{r, echo=FALSE} library("rmarkdown") render("Introduction-to-R.Rpres",output_format =
"pdf_document",output_file = 'Introduction-to-R.Rpres.pdf')
```