Data Management and Visualization

Lecture 1: Introduction

Ekarat Rattagan, PhD
IST, MUT
1/2561

CV

- **Name**: Ekarat Rattagan

- **Current work**:
  - Lecturer at Faculty of Information Science and Technology, Mahanakorn University of Technology, Thailand

- **Education**:
  - Ph.D., Electrical Engineering and Computer Science, National Chiao Tung University (NCTU), Taiwan
  - M.Sc., Information Technology, King Mongkut's University of Technology Thonburi (KMUTT), Thailand
  - B.Arch, Architecture, Chulalongkorn University (CU), Thailand

- **Research interest**:
  - IoT, Mobile system and application, Interactive system, AI, Video game design and development

- **Work experience**:
  - Software engineer, Embedded Benchmarking Lab (EBL), Taiwan
  - Video Game developer, Novaleaf Software, Thailand
  - Software Engineer, Incotec-automation, Thailand

- **Selected published papers**:
  - "Symbolic Regression and Clustering for Power Consumption Estimation on Smartphone Hardware Subsystem," ***Taiwan patent***, 2015
  - "Accurate Traffic Flow Prediction in Heterogeneous Vehicular Networks in an Intelligent Transport System Using a Supervised Non-Parametric Classifier," ***Sensors,*** 2018 (IF 2.6)
  - "Clustering and Symbolic Regression for Power Consumption Estimation on Smartphone Hardware Subsystems", ***IEEE Transactions on Sustainable Computing***, 2018 (New journal)
  - "Semi-online Power Estimation for Smartphone Hardware Components", ***IEEE Transactions on Sustainable Computing***, 2016 (New journal)
  - "Calibrating Parameters and Formulas for Process-level Energy Consumption Profiling in Smartphones", ***Journal of Network and Computer Applications***, 2014 (IF 3.4)

- **Service**:
  - Publicity chair for IEEE-SSCI 2018.
  - Journal Reviewers, IEEE Tx on Sustainable Computing, IEEE embedded system letters, Journal of Information Science and Engineering,

Syllabus

ครั้งที่	เรื่อง
1	Course overview, Review R programming
2	Data management I: data collection
3	Data management II
4	Data management III
5	Data visualization I: Filtering & Aggregate
6	Data visualization II: Perception, Cognition, Colors
7	Data visualization III: Visualization data
สอบกลางภาค	
8	Designing visualization
9	Visualization views
10	Visualization Tables
11	Visualization Graphs
12	Map
13	Spatial visualization
14	Animation
15	Real world case study, e.g., Airbnb, Agoda, Uber
สอบประจำภาค	

Scores

- **Midterm exam 30%**
  - Contents —> (Lecture + Lab) (Week 1 ~ 7) + 1 assigned papers)
- **Final exam 30%**
  - Contents —> (Lecture + Lab) (Week 8 ~ 15) + 1 assigned papers)
- **Project 30%**
  - Idea presentation (Week 5)
  - Final presentation (Week 15)
- **Class attending 10%**

Three questions for learning this course

- What
- Why
- How

Data Management

Definition: The development and execution of architectures, policies, practices and procedures that properly manage the full data lifecycle. [DAMA]

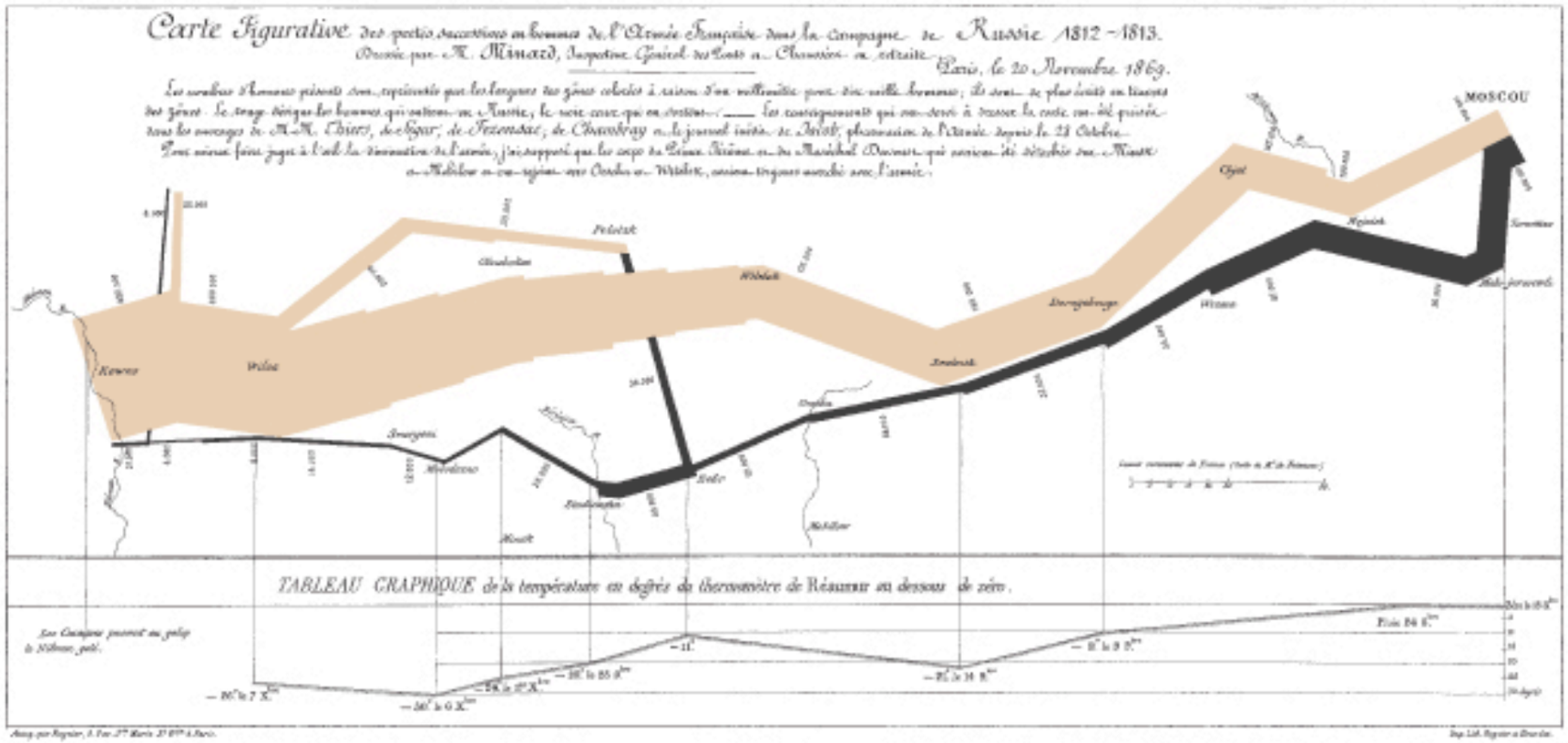
- **Data collection**
    - Ability to get to and retrieve information
- **Data quality**
    - Making sure data is accurate and usable
- **Data integration**
    - Combining different types of data
- **Data streaming**
    - Analyzing data as it moves, e.g., realtime data

Data Visualization (DataViz)

DataViz: How can I see what my data show?

- creating appropriate and informative pictures of a dataset
- iterative exploration of data

Napoleon's March



Napoleon's March to Moscow The War of 1812

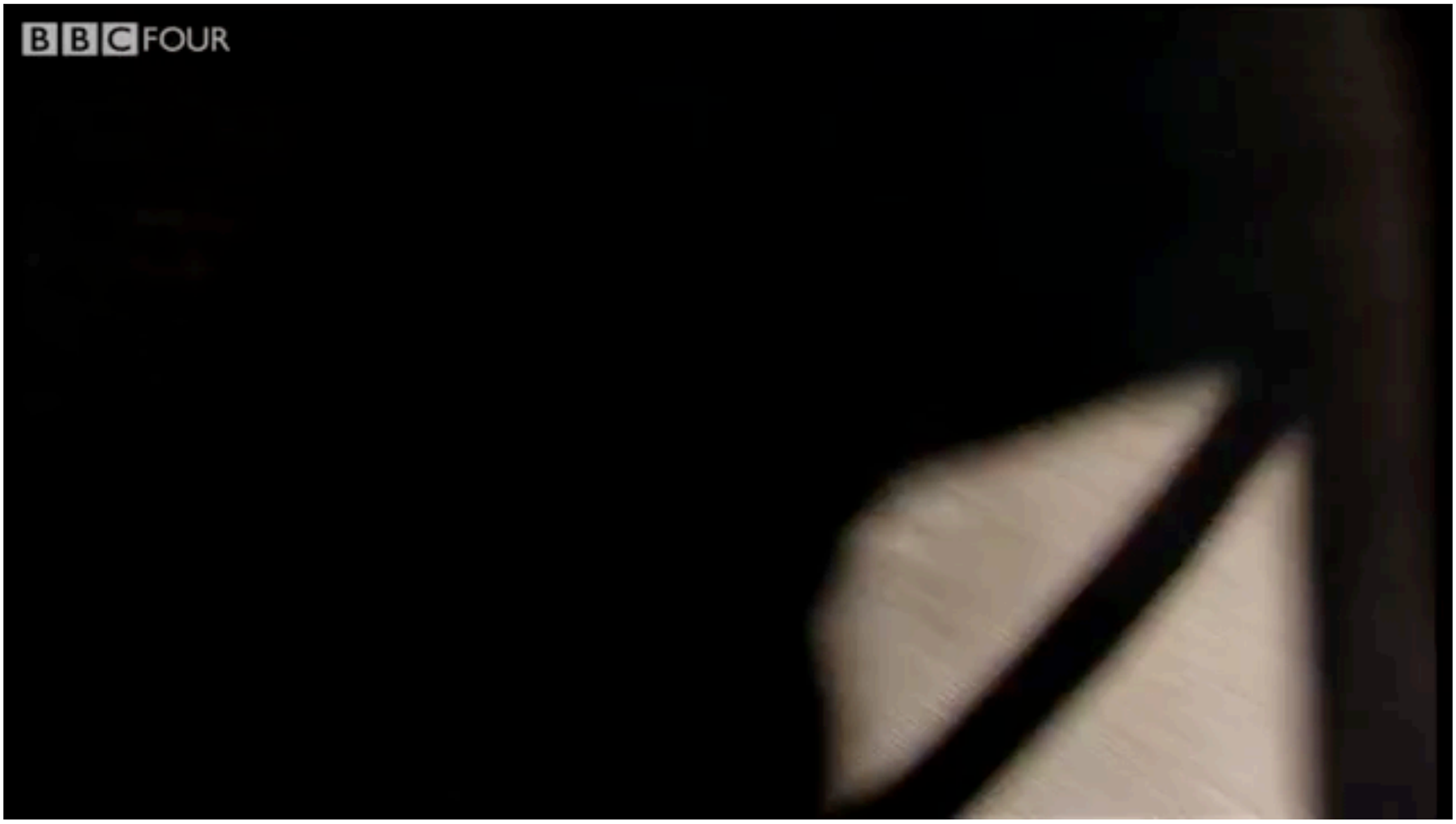
Charles Joseph Minard

This classic of Charles Joseph Minard (1781–1870), the French engineer, shows the terrible fate of Napoleon's army in Russia. Described by E. J. Marey as seeming to defy the pen of the historian by its brutal eloquence, this combination of data map and time-series, drawn in 1869, portrays the devastating losses suffered in Napoleon's Russian campaign of 1812. Beginning at the left on the Polish-Russian border near the Niemen River, the thick band shows the size of the army (422,000 men) as it invaded Russia in June 1812. The width of the band indicates the size of the army at each place on the map. In September, the army reached Moscow, which was by then in ruins and deserted, with 100,000 men. The path of Napoleon's retreat from Moscow is depicted by the darker, lower band, which is linked to a temperature scale and dates at the bottom of the chart. It was a bitterly cold winter, and many froze on the march out of Russia. As the graphic shows, the crossing of the Berezina River was a disaster, and the army finally straggled back into Poland with only 10,000 men remaining. Also shown are the movements of auxiliary troops, as they sought to protect the rear and the flank of the advancing army. Minard's graphic tells a rich, coherent story with its multivariate data, far more enlightening than just a single number bouncing along over time. Six variables are plotted: the size of the army, its location on a two-dimensional surface, direction of the army's movement, and temperature on various dates during the retreat from Moscow. It may well be the best statistical graphic ever drawn.

Edward R. Tufte, *The Visual Display of Quantitative Information* Graphics Press Box 430 Cheshire, Connecticut 06410

According to Tufte: "It may well be the best statistical graphic ever drawn."
5 variables: Army Size, location, dates, direction, temperature during retreat

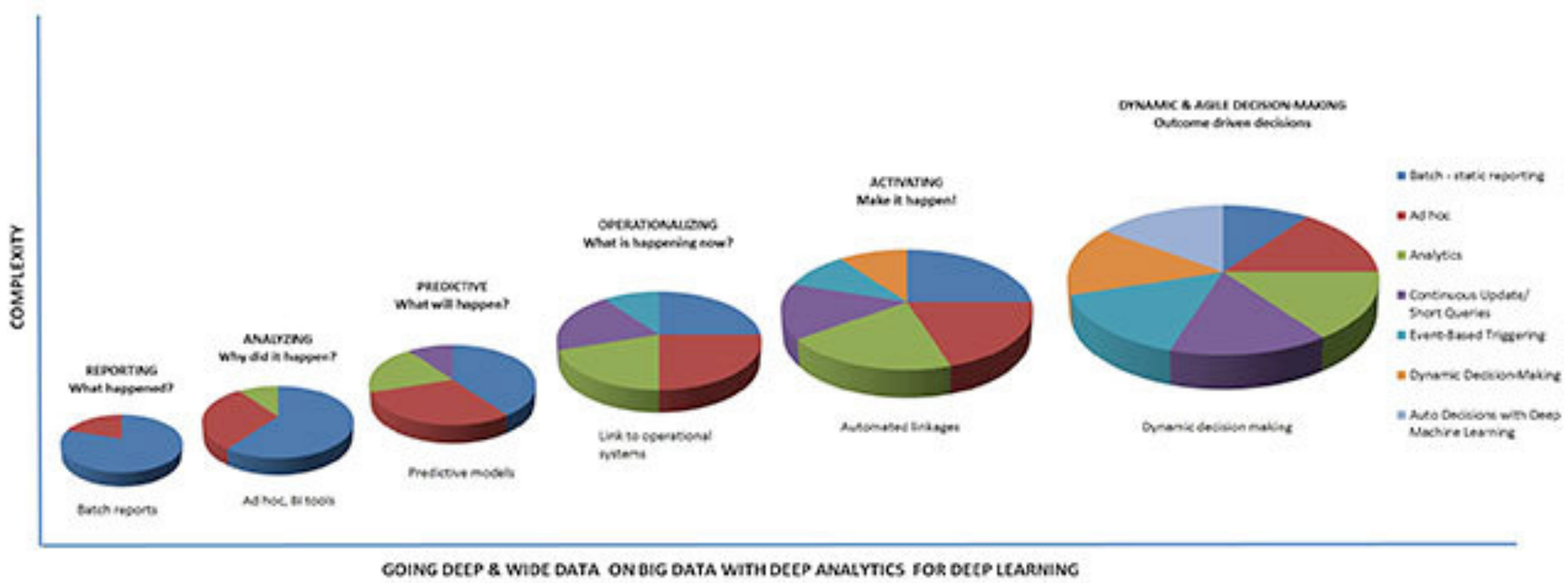
Good examples (I)



Good examples (II)

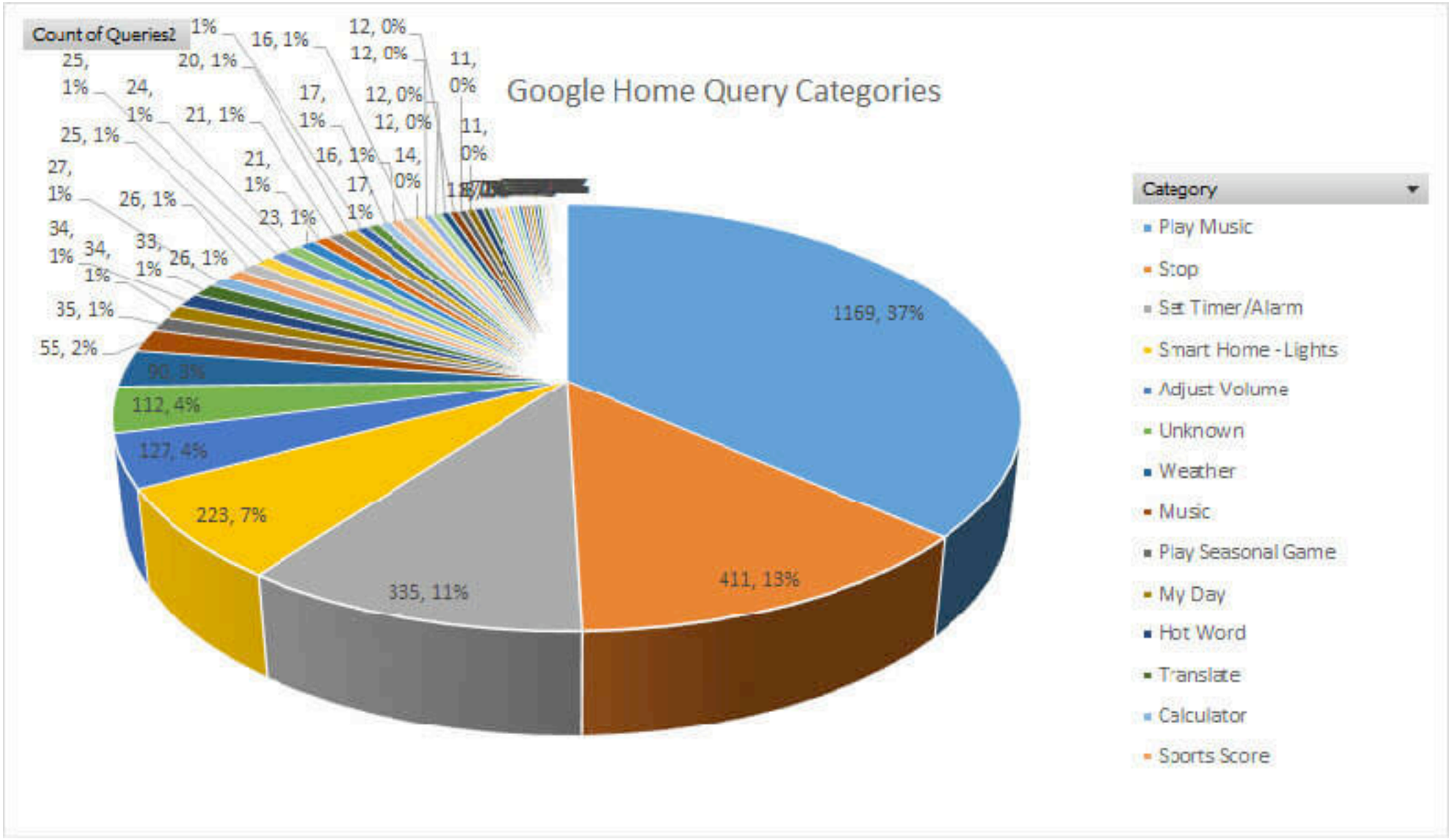
- NY Times Interactive Visualizations (e.g., 2013 Federal Budget)
    - http://www.nytimes.com/interactive/2012/02/13/us/politics/2013-budget-proposal-graphic.html

May be good examples after revision (I)



http://viz.wtf/

May be good examples after revision (II)



http://viz.wtf/

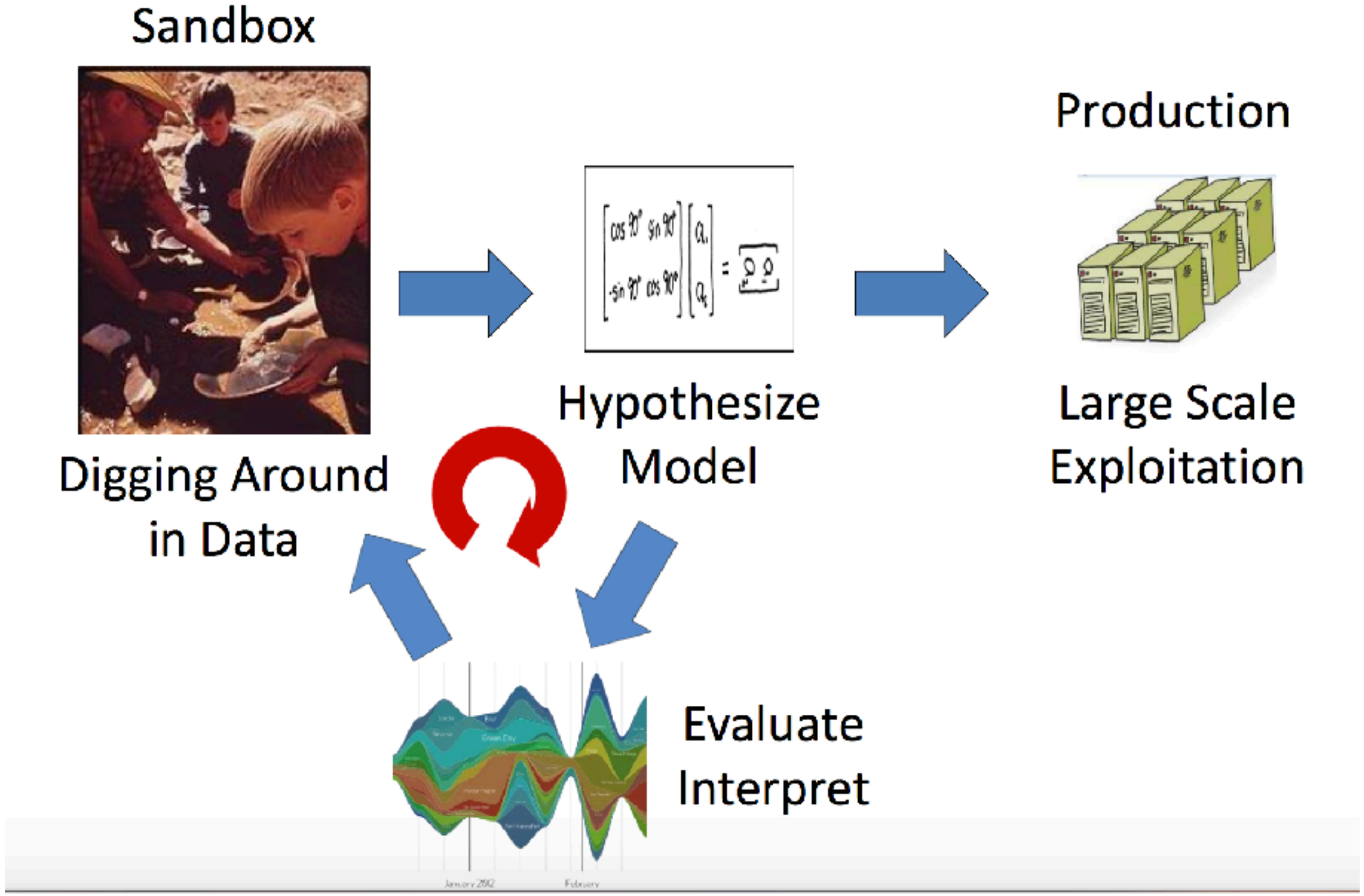


Image captured from CS 194 Fall 2014

A part of Data Scientist's Workflow

DataViz used for

- **Analysis:** Reasoning about information
  - Finding relationships
  - Discover structure
  - Quantifying values and influences
  - Should be part of a query/analyze cycle

- **Communication:** Inform and persuade others
  - Capture attention
  - Tell a story visually
  - Focus on certain aspects, and omit others

Three principles for visualization

1. **Be true to your research** — design your display to illustrate a particular point

2. **Maximize information, minimize ink** — use the simplest possible representation for the bits you want to convey

3. **Organize hierarchically** — what should a viewer see first? what if they look deeper?

Basic R

The Comprehensive R Archive Network (*CRAN*)

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the CRAN Repository Policy and then use the web form.

If this fails, upload to ftp://CRAN.R-project.org/incoming/ and send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for all binaries of their platform, respectively.

https://cran.r-project.org/

Data Types

Numbers

- a <- 3 #3
- b <- sqrt(a*a+3) #3.464102
- ls ( ) # get a list of defined variables, “a” “b”
- a <- numeric(10) # Initial ten ‘0’ number
- typeof(a) # double
- a <- 1:5 # 1 2 3 4 5

Data Types

Strings

- a <- “hello” #“hello”
- b <- c(“hello”, “there”) # “hello” “there”
- b[1] #”hello”
- typeof(a) # “character”

Logical

- a <- TRUE
- a <- c(TRUE, FALSE)

Operators

Arithmetic Operators

Operator	Description
+	addition
-	subtraction
*****	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Logical Operators

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Data Types

Vectors & Assignment

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
> a[1]
1
> a[0]
numeric(0)
> a[7]
NA
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE) #logical vector
```

Matrices

```
# generates 5 x 4 numeric matrix
x <- matrix(1:20, nrow=5, ncol=4)
x[,4] # 4th column of matrix
x[3,] # 3rd row of matrix
x[2:4,1:3] # rows 2,3,4 of columns 1,2,3
```

Data Types

Lists: an ordered collection of objects. Allow you to gather a variety of (possibly unrelated) objects under one name.

```
# example of a list with 4 components -
# a string, a numeric vector, a matrix, and a scaler
my_list <- list(name="Fred", my_numbers=a, my_matrix=x,
age=5.3)
```

Identify elements of a list using the [[]] convention.

```
my_list[[2]] # 2nd component of the list
my_list[[“my_numbers”]] # component named my_numbers in list
```

Data Types

Arrays: contain any number of dimensions, while a matrix is a 2-dimensional array.

```
# 2D
x <- array(1:20, dim=c(4,5))
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

```
#3D
x <- array(1:20, dim=c(2,5,2))
> x
, , 1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
, , 2
     [,1] [,2] [,3] [,4] [,5]
[1,]   11   13   15   17   19
[2,]   12   14   16   18   20
```

Data Types

Factors: A factor is a vector object used to specify a discrete classification (grouping) of the components of other vectors of the same length.

```
> state <- c("tas", "sa",  "qld", "nsw", "nsw", "nt",  "wa",  “wa", "qld", "vic", "nsw", "vic", "qld", "qld", "sa",  "tas",
                "sa",  "nt",  "wa",  "vic", "qld", "nsw", "nsw", “wa", "sa",  "act", "nsw", "vic", "vic", “act”)
> statef <- factor(state)

> statef

[1] tas sa  qld nsw nsw nt  wa  wa  qld vic nsw vic qld qld sa  tas sa

[18] nt  wa  vic qld nsw nsw wa  sa  act nsw vic vic act

Levels: act nsw nt qld sa tas vic wa
```

To find out the levels of a factor the function levels() can be used.

```
> levels(statef)

    [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" “wa"

> incomes <- c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56,
                61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46,
                59, 46, 58, 43)

> incmeans <- tapply(incomes, statef, mean) giving a means vector with the components labelled by the levels
       act    nsw     nt    qld     sa    tas    vic     wa
    44.500 57.333 55.500 53.600 55.000 60.500 56.000 52.250
```

Data Types

Data Frames

Different columns can have different modes (numeric, character, logical, etc.).

```
d <- c(1,2,3,4)
e <- c("red", "white", "red", NA)
f <- c(TRUE,TRUE,TRUE,FALSE)
myData <- data.frame(d,e,f)
names(myData) <- c("ID","Color","Passed") # variable names
```

There are a variety of ways to identify the elements of a data frame .

```
myData[3:5] # columns 3,4,5 of data frame
myData[c("ID","Age")] # columns ID and Age from data frame
myData$ID # variable ID in the data frame
```

Read/Write data

1. scan: Read data into a vector or list from the console or file.

```
> x <- scan()
1: 3 5 6
4: 3 5 78 29
8: 34 5 1 78
12:
Read 11 items
```

```
> y <- scan(what=" ")
1: red blue
3: green red
5: blue yellow
7:
Read 6 items
```

```
# inputting a text file and outputting a list
(x <- scan("https://raw.githubusercontent.com/pokekarat/Course/master/Data_Visual/scan.txt",
           what = list(age = 0, name = "")))
## $age
## [1] 12 24 35 20
##
## $name
## [1] "bobby"   "kate"    "david"   "michael"
```

2. read.table: Read data into data frame.

```
myData <- read.table("c:/test.csv", header=TRUE, sep=",")
```

* Note that read.csv is a fairly thin wrapper around read.table

3. Write data

```
write.csv(myData, file = "test.csv")
```

Loop

A loop is a way to repeat a sequence of instructions under certain conditions.

1. For

```
for (variable in sequence) {
    expression
    expression
    expression
}
```

Example

```
> for (x in c(1:10)) print(sqrt(x))
[1] 1
[1] 1.414214
[1] 1.732051
[1] 2
[1] 2.236068
[1] 2.449490
[1] 2.645751
[1] 2.828427
[1] 3
[1] 3.162278
```

2. While

```
while (condition) {
    expression
    expression
    expression
}
```

```
a <- 0
b <- 1
print(a)
while (b < 50) {
    print(b)
    temp <- a + b
    a <- b
    b <- temp
}
```

User-written functions

```
myfunction <- function(arg1, arg2, ... ){
    statements
    return(object)
}
```

Example

```
my_mean <- function(x) {
    center <- mean(x);
    cat("Mean=", center, "\n")
    #return(center)
}

> my_mean(5)
Mean= 5
[1] 5
> my_mean(1:5)
Mean= 3
```

Functions

Numeric Functions

Function	Description
abs(*x***)**	absolute value
sqrt(*x***)**	square root
ceiling(*x***)**	ceiling(3.475) is 4
floor(*x***)**	floor(3.475) is 3

Statistical Functions

Function	Description
mean(*x***, trim=**0**, na.rm=**FALSE**)**	mean of object x # trimmed mean, removing any missing values and # 5 percent of highest and lowest scores
sd(*x***)**	standard deviation of object(x). also look at var(x) for variance and mad(x) for median absolute deviation.
median(*x***)**	median

Other Useful Functions

Function	Description
seq(*from* **,** *to***,** *by***)**	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)
rep(*x***,** *ntimes***)**	repeat *x n* times y <- rep(1:3, 2) # y is c(1, 2, 3, 1, 2, 3)

Functions

`paste` is concatenate a series of strings.

```
# First example
paste("file", "number", "32")
[1] "file number 32"
```

```
# Second example
paste("file", "number", "32", sep = "_")
[1] "file_number_32"
```