Why Scala for Data Science?

Nathaniel Osgood
University of Saskatchewan

Data Science

- "Data Science" seeks to provide systems, methodologies and procedures for deriving insight from data
- Much of data science current focuses on processing and making sense of "big data"

Some Sources of “Big Data”

- Twitter (feeds)
- Facebook (status updates)
- Environmental sensors (weather, municipal, building)
- Lab test results
- Point of sale and online sales records
- Administrative data
- Questionnaire responses (mobile, web)
- Sequence data
- Supply chain data feeds
- Voice audio
- Incoming/outgoing calls
- Communication infrastructure proximity data
- Health information browsing behavior
- Consumer electronic devices sensors (physical activity, proximity, location, etc.)

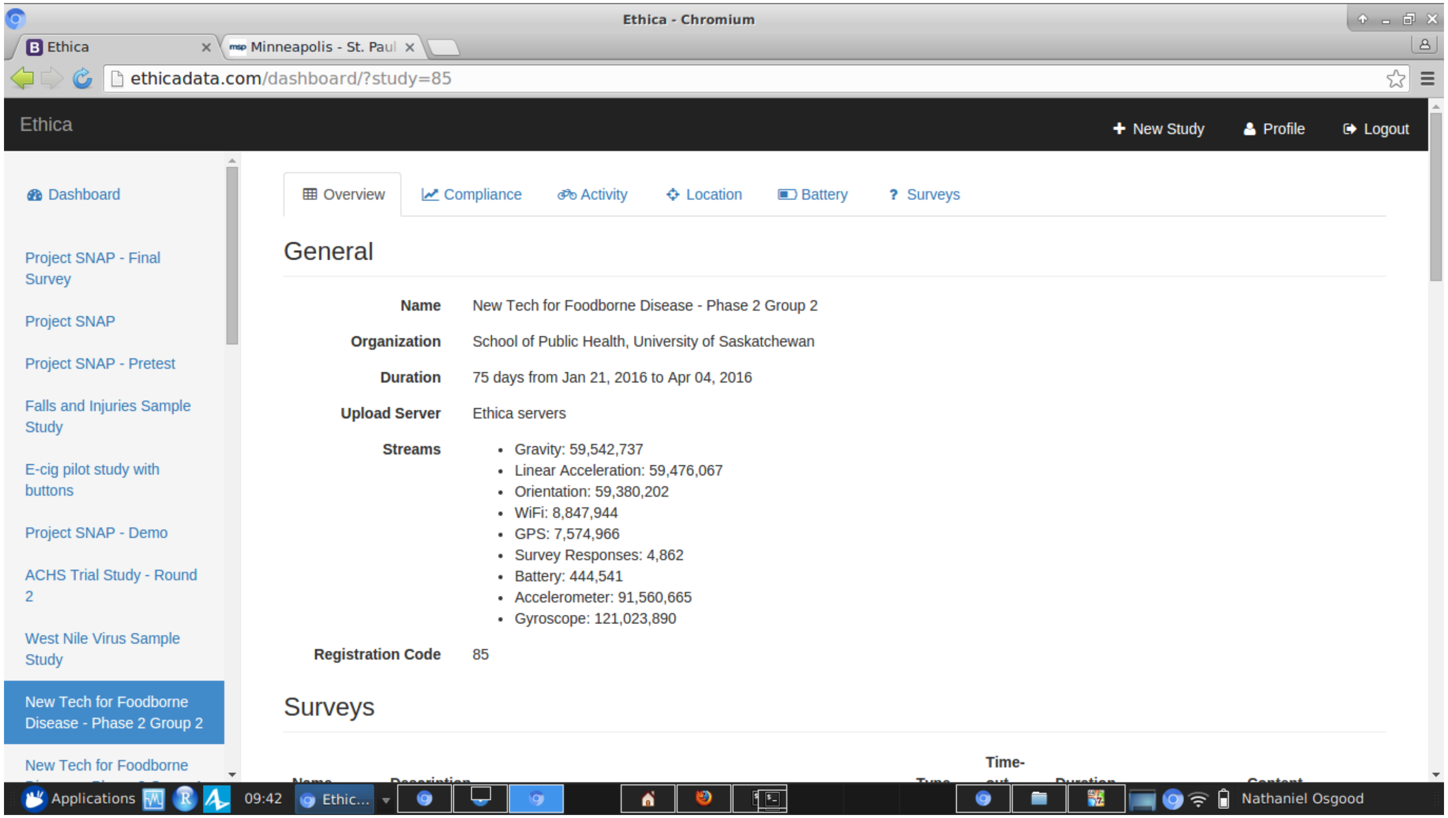
Four key “V’s” of “Big Data” (Google)

- **Volume**: Lots of evidence
- **Velocity**: High temporal resolution longitudinal data
- **Variety**: Cross-linked data sources support can “triangulation” of understanding
- **Veracity**: Physical measures are less subject to self-report, on-device self-reporting is more temporally proximate to phenomena of interest event (exposures, symptoms,…)

Volume

- Consider
  - N participants
  - # of records per participant (M)
- Traditional epidemiologic studies: N >> M
- “Big data”: M >> N common
- Common: Dozens of MB per participant/day
- This volume of data will often require different handling techniques than for traditional systems: Different
  - Storage
  - Analysis
  - Visualization

Volume & Variety: Some Statistics



Velocity

- Electronic data sources often update frequently
  - Low rates: Lab data, administrative data
  - Medium Low rate: multiple times/day e.g.,
    - Facebook updates
    - Twitter
    - browsing behavior
    - app use
    - Ecological momentary assessment (EMA) responses
    - Weather
    - Point of sale
  - Medium rate: on order of seconds (e.g., GPS, building sensors)
  - Higher rates: Many times per second (e.g., accelerometers, gyroscopes)
- Such velocity provides high temporal resolution into micro-behaviours and exposures

Variety

- A given electronic data sources often provide multiple lines of evidence
  - Smartwatch (e.g., Empatica E4): stress responses via electrodermal activity & Heart Rate Variability, heart rate, acceleration, skin temperature
  - Smartphone with Ethica iEpi: location, physical activity, proximity, posture, humidity, EMA responses, etc.
- For a given participant, we increasingly have multiple sources of electronic data available – both quantitative and qualitative
  - Smartphone (context and state via sensors, ecological momentary assessments)
  - Smartwatch
  - Weather
  - Point of sale
  - Facebook updates
- This evidence is **cross-linked** by **participant** and **time** (i.e., for a given participant & time, we can find the relevant information applying then across all data sources
- We can often **triangulate** state of a given participant using many lines of evidence

Need for *Scalability* in Data Science

- Very large amounts of data, but limited resources
  - Memory (often data far outpaces physical memory)
  - CPU speed
- Opportunities for speed come from using many computational resources rather than speeding up our processors directly
- Capacity to robustly handle exceptional situations while conducting large-scale processing
  - Failure across machines
  - Missing values (e.g., NAs)
  - Error conditions in processing (e.g., divide by 0)
- Translating solutions readily across spectrum of needs

Why Scala?

- Support for functional programming benefits (next slide)
- Multiple key Data Science needs supported (some via FP)
  - Much data, Limited MEMORY => want to avoid materializing data structures => **laziness, recomputation**
  - Much data to process => want to use many processors concurrently to handle => **clear dependencies, parallel data structures, parallelizable higher order functions**
  - Capacity to robustly handle exceptional situations -- even for parallel code => **Type-checked handling of errors, missing values, failure**
  - Scalable from data exploration to large scale rigour
  - Data exploration => need for flexibility in processing => modular pipelines of operations
- Rich type system supporting subtyping and parameterized types
- Mix of
- Compiled
  - Rigorous static type checking

Benefits of Functional Programming

- Equational reasoning => transparency in understanding code
- Composition of simple mechanisms yield powerful results
- Modular -- orthogonality via higher order functions
- More transparent error handling, clear values
- Clear dependencies
- Easier means of specifying asynchronous operations
- In theory (but less clear at a practical level): potential for pipelining
- Ability to send code to different machines with minimal trouble
- Immutability rules out many common errors
- Support for transparent, simple-to-use higher-order functions to process in ways that can be parallelized/vectorized