Analysis - Overview

Analysis Overview
Overview of Material

- How do we analyse qualitative data?
  - What type of analysis
  - Elements of analysis
  - Inductive and Deductive analysis
- Grounded Theory
  - One method for analysing qualitative data

Analysis Overview
Objective

- Fieldwork produces
  - Voluminous raw data
- Analysis turns this raw data into findings
  - Search for patterns in data
  - Ideas that help explain why those patterns are there
- Analysis process
  - Organise fieldnotes into readable narrative descriptions
  - Major themes/categories identified
  - Illustrative case studies provide

Analysis Overview
A quirk of grammar

- Qualitative Data Analysis
  - Qualitative Analysis of Data (what kind of data?)
  - Analysis of Qualitative Data (what kind of analysis?)
- Four possibilities
  - Qualitative Analysis of Qualitative Data
    - This is what we will primarily focus on
  - Qualitative Analysis of Quantitative Data
    - The search for meaning of quantitative data (in some senses there's always some qualitative analysis of quantitative data
  - Quantitative Analysis of Qualitative Data
    - Turning data from words and images into numbers, "descriptive statistics"
  - Quantitative Analysis of Quantitative Data

Analysis Overview
Being the Instrument Again

- You are the analytic instrument
  - So once again the question of bias creeps in
    - Are you really seeing those patterns, or are you wanting to see things
- As in data collection
  - Triangulation: using multiple sources to confirm event
  - Alternate explanations: seek others in field and colleagues and ask them for alternate explanations of your data
    - In some disciplines data analysis is a collaborative effort to avoid single person biases
  - Embrace data that doesn't fit in
    - Challenges your analysis to be broader than it would otherwise be

Analysis Overview
Process

- Analysis begins before data collection
  - You have some ideas about what you are going to study before you start
  - Why the things you are going to study matter
  - This is analysis
- Analysis cycles with data collection
  - During data gathering, you'll be analysing what you see and hear
    - Analytic notes that appear in your field notes
  - You'll periodically stop data collection, conduct analysis
    - Analysis will likely yield questions or gaps in your knowledge which focus next collection
  - Full and final analysis after data collection
    - Final synthesis and written report

Analysis Overview
Three Elements Common to All Analysis

- Data reduction
  - The process of selection, focusing, simplifying, abstracting the raw data
    - Going on throughout the qualitative research study
- Data organisation
  - The process of organising the reduced data in ways that allow you to begin to generate explanations
    - Little of this early on, gets more significant throughout the study
- Data explanation and verification
  - Drawing conclusions from the explanations (organising the explanations)
    - Most intensive at the end
  - Testing the conclusions drawn: verifying their plausibility

Georgia Institute of Technology

Analysis
Inductive or Deductive

- Inductive Analysis
  - Let the analytic themes emerge from the study of the data
  - Happens in the exploratory phases of analysis
    - Early to middle phases of explanation development
- Deductive Analysis
  - Starting with a hypothesis for data analysis
  - Happens in the confirmatory stage of analysis
    - Middle to late phases of explanation development and confirmation
- Maybe conducting both Inductive and Deductive
  - In one study because some concepts further developed than others

Georgia Institute of Technology

Grounded Theory
One Approach to Data Analysis

Grounded Theory
Overview

- What is it?
- How do I do it?
- Concepts
  - Theoretical Sensitivity
  - Open Coding
  - Axial Coding
  - Selective Coding
- Illustrated with an example
  - From my own research!

Grounded Theory
What is it?

- An approach to guide analysis of data
  - The Discovery of Grounded Theory: Strategies for Qualitative Research
  - Barney G. Glaser and Anselm L. Strauss
- Set of techniques
  - Identify categories and concepts that emerge from text
    - Data reduction in our terminology
  - Link concepts into substantive and formal theories
    - Data organisation and explanation in our terminology

Georgia Institute of Technology

Grounded Theory
Overview of the Mechanics

- Read through field notes
  - Produce *analytic categories*—potential themes that arise
- As categories emerge
  - Pull data from categories together and compare them
- Think about how categories
  - Fit together into an explanation, or model
- Take models developed
  - Check them against the data—particularly negative cases
- Present results using examples from the data

Georgia Institute of Technology

Grounded Theory
Getting Started

- Before you commence analysis
  - Theoretical sensitivity is about developing insight into data
    - Being able to give the data meaning, understand it, and separate pertinent from not
- Theoretical sensitivity
  - Use literature: readings on theory, research, and supporting evidence
    - Ground yourself in the multiple contexts that surround the data your analysing
  - Professional experience
    - Use background knowledge of practioners in the field (including yourself)

Grounded Theory
Example: Theoretical Sensitivity in Software

- Prior to analysis of start-up data
  - I did the following things to enhance my theoretical sensitivity
- Literature: software engineering
  - Followed ICSE proceedings, SCM workshop etc. to understand CM
- Professional experience: get others insights
  - Followed CM usenet group to learn about practice of CM manager

Grounded Theory
First Step: Open Coding

- Open coding
  - Process of breaking down, examining, comparing, conceptualising and categorising data
    - Data reduction (although it doesn't feel like "reduction" when you're doing it)
    - Inductive
- Open coding consists of
  - Labelling phenomena
  - Discovering categories
  - Developing categories: properties and dimensions

Georgia Institute of Technology

Open Coding
Labelling Phenomena

- Break down raw full descriptive field notes
  - By asking questions about notes
    - What is this?
    - What does it represent?
  - For each phenomenon (incident, idea or event)
- Give each discrete phenomenon a name
  - Give it a name that captures its essence in a more general way: *concept*
- Compare it to others already discovered
  - Could it be labelled in a way that others are labelled preserving integrity

Georgia Institute of Technology

Open Coding
Discovering Categories of Concepts

- At the end of even a small piece of field notes
  - Lots of concepts
    - It is quite normal to feel overwhelmed and concerned at this point
- Next, group concepts into *categories*
  - Category: classification of concepts, discovered when concepts compared
    - What's similar (may want to note why you think it's similar)
  - This started when you labelled
  - Now you're asking questions about the concepts and the category
    - What are the phenomena in this category about, what are they instances of
  - Use answer to label category

Georgia Institute of Technology

Open Coding
Developing Properties and Dimensions

- Properties
  - Characteristics or attributes of a category
    - Eg: colour has the properties of intensity and hue
- Dimensions
  - Locations of a property along a continuum
    - Eg: intensity varies from high to low, hue from darker to lighter
- These will help us develop broader relations later

Georgia Institute of Technology

Example
Open Coding

- Context
  - Interview with a software developer about her work

Well I try to avoid parallel development, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by keeping things separate as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to merge these two together, and merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

Example: Open Coding
Phenomena

Well I try to avoid **parallel development**, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by keeping things separate as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to **merge** these two together, and merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

- Phenomena: events being described
  - Parallel development
  - Merge
- Picked these because they seemed to be main acts

Example: Open Coding
Category Discovery and Development

Well I try to avoid **parallel development**, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by *keeping things separate as far the units of work, the resolutions of work*, which in our case is source files, and number 2 when you go about *assigning this work* you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to **merge** these two together, and merging to be honest is generally pretty easy, as long as the people aren't *working on the same checks in the code*. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

- Category
  - Initially, code coordination, but too broad so *Individuals coordinating code*
- Properties and Dimensions
  - *Work*: that varies from independent to dependent
    - What's being talked about here? Work talked about -- see italics
  - *Module Change*: varies from separate to the same
    - Changes to the code, modules specifically, described in detail

Open Coding
Practical Moment

- Computers probably useful for open coding
  - Sadly, I've never done that way—I'm still manual
- Initially I used post-it notes for each concept
  - And a large flat surface
    - My slide door wardrobe which I then could not get into for several days
- Now I tend to use absurd numbers of 3x5 cards
  - And a floor or conference room table
  - One colour for concepts, one concept per card
  - Another colour for categories, one category per card

Grounded Theory
Second Step: Axial Coding

- Axial Coding
  - Data assembled in new ways after open coding, by making connections between categories
    - Moving from inductive to deductive analysis
- Axial Coding consists of
  - Taking categories and identifying
    - The *conditions* that give rise to it
    - *Context* into which it is embedded
    - *Action/interaction strategies* in which it is handled, managed, carried out
    - *Consequences* of those strategies
  - Expanding out our knowledge of the categories
- Categories represent our "phenomena" now

Georgia Institute of Technology

Axial Coding
Casual Conditions and Context

- Causal Conditions
  - Events, incidents that lead to the occurrence of a category
    - Search the data for the conditions that give rise to the category phenomena
- Context
  - The set of properties that pertain to a category
    - Locate the events in the category on the dimensions

Georgia Institute of Technology

Axial Coding

Intervening Conditions and (Inter)Actional Strategies

- Intervening Conditions
  - Broader structural context pertaining to category
    - Space, time, culture, economic status, technological status, career, history, biography
- Action/Interactional Strategies
  - Particular emphasis of Grounded Theory: it focuses on action & interaction
    - What actions do individuals take with respect to the category
    - How do groups or collectives interact and act with respect to the category

Axial Coding

Consequences

- Consequences
  - There are always consequences :-)
  - Here it means the outcomes of the (inter)actional strategies

Example: Axial Coding
Casual Conditions

Well I try to avoid parallel development, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by keeping things separate as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to merge these two together, and merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

- *Individuals coordinating code*
- What causes individuals coordinating code?
  - bad management
  - bad code
  - being in the same part of the code

Example: Axial Coding
Context

Well I try to avoid parallel development, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by keeping things separate as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to merge these two together, and merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

- *Individuals coordinating code*
  - *Work*: that varies from independent to dependent
    - Dependent work
  - *Module Change*: varies from separate to the same
    - Same changes

Example: Axial Coding
Intervening Conditions

Well I try to avoid parallel development, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by keeping things separate as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to merge these two together, and merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

- *Individuals coordinating code*
- What broader contexts might apply here?
  - Time: delivery schedules
  - Space: colocated development or geographically separated
- At this point, with this data
  - These are questions we might explore in the field

Example: Axial Coding
(Inter)actional Strategies

Well I try to avoid parallel development, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by *keeping things separate* as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to merge these two together, and merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code. If I'm *working at the top of the file and somebody else is working on something and the bottom of the file* then it's fairly easy to merge unless those changes change the overall algorithm, then it gets messy.

- *Individuals coordinating code*
- Strategies focus on avoiding interaction
  - Example: waiting for someone else to check in before she checks out
- Here strategies emerge in negative form
  - What not to do

Example: Axial Coding
Consequences

Well I try to avoid parallel development, I grumble about it, to me it's out there, it happens in our company and in others, but it seems to me that if there's better management and better decomposition of problems then should be avoided. Number 1 solve it by keeping things separate as far the units of work, the resolutions of work, which in our case is source files, and number 2 when you go about assigning this work you could try and assign common problems to the same person so they are not doing parallel development. ... What has to happen is the last guy who checks something in has to merge these two together, and *merging to be honest is generally pretty easy, as long as the people aren't working on the same checks in the code*. If I'm working at the top of the file and somebody else is working on something and the bottom of the file then it's fairly easy to merge unless those changes change the overall algorithm, then it gets *messy*.

- *Individuals coordinating code*
- Not keeping things separate
  - Leads to a mess (another negative example)
- Separation leads to ease
  - Of interaction (positive example)

Grounded Theory
Second Step: Selective Coding

- Selective coding
  - The process of selecting the core category
    - The central category around which your final analysis will be based
  - Then relating it to other categories
- Three processes
  - Explicating the story line: analytic description of the core category
  - Relating other categories to the core
  - Validating the story line

Selective Coding
The Story Line

- Commit, commit commit!
  - After months of analysis EVERYTHING seems important
  - But a story line needs to priortise one category over all others
- Finding the story
  - Ask yourself what seems the most striking/interesting
    + Write it in no more than a paragraph, just a FEW sentences
- Does one category seem more central?
  - Can it explain the others?
  - If you have two, drop one!

Selective Coding
Relating Categories to the Story Line

- First, outline core's properties and dimensions
- Relate other categories to the core
  - Using properties and dimensions
  - Using conditions, context, strategies, consequences

Selective Coding
Validating the Story Line

- Final step
  - Validate the theory against the data
- Write a series of memos that step through the story
  - Do they organise all the data
  - Do they cope with all the negative as well as positive examples
- Go back to the field
  - To fill in missing pieces

Example

- Previously, we had one category
  - Individuals coordinating code
    - Parallel development, history of the code base ,many others
- Others emerged
  - Groups coordinating code
    - Life cycle, architecture
  - Organisations coordinating code
  - Multiple organisations coordinating code
- And these are just the relevant ones
  - For the gory details, there's my dissertation!

Example: Selective Coding
The Story Line

- Eventually the story line became
  - Technical dependencies in software create social dependencies among the developers responsible for the pieces
    + Named the core category: recomposition
    + Recomposition is all the work it takes to coordinate the assembling software from its parts
    + Name came from decomposition, the process of breaking software apart
- It took me a long time to abandon
  - Individual, group, organisational distinction
  - Present in categories, but not actually important to the central story
    + Important in explaining the details of the theory, but not in summarising it

Example: Selective Coding
Relating Categories to the Story Line

- Recomposition
  - Seemed to encompass individual, group, organisational coordination
  - Same roots
- Properties of interrelationship of work and code
  - All the problems happen when work and code can not be separated
- Differences emerged
  - Causes and consequences depending on scale—individual v. organisation
    + But they could be parameterised and organised into the story line

Example: Selective Coding
Validating the Story Line

- Revisited my data with this analytic focus
  - Wrote my dissertation using the core category story line
    - Good story line—explanation—is a thesis
- Other ways to validate
  - Go into other settings and see whether it works
- Publish it
  - Independent confirmation of ideas

Grounded Theory
Theories and Resources

- Use of theories in grounded theory
  - Can use grounded ones
  - So, for example can use articulation work in your own analysis
    - As a starting point for organising the analysis
- Books for the interested
  - Qualitative Analysis for Social Scientists
  - Basics of Qualitative Research

Analysis
Summary

- Overview of Analysis
  - What is it and what does it involve
- Introduction to Grounded Theory
  - One method for analysing qualitative data
  - Theoretical Sensitivity
  - Open, Axial, and Selective Coding
    - With an example

Georgia Institute of Technology