Preserving Type I error in rare allele GWAS

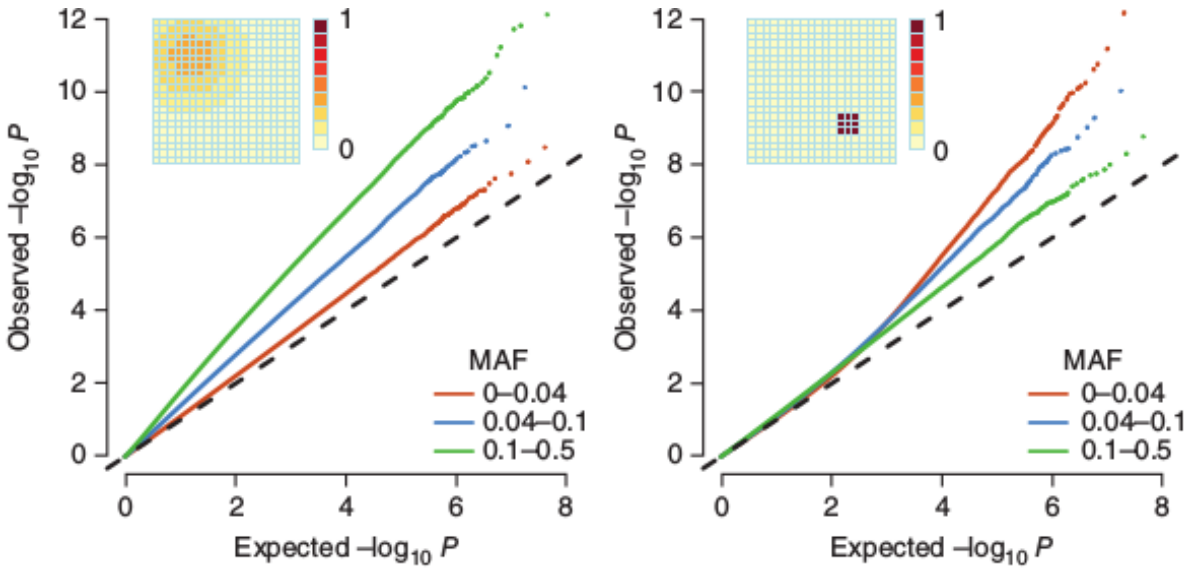
Dan Schlauch, PhD Candidate

Department of Biostatistics
Harvard School of Public Health

November 5, 2015

Outline

Rare allele association inflation



QQ plots of p-values separated by allele frequency. Comparing two types of non-genetic risk distributions. **Mathieson, Nature Genetics 2012**

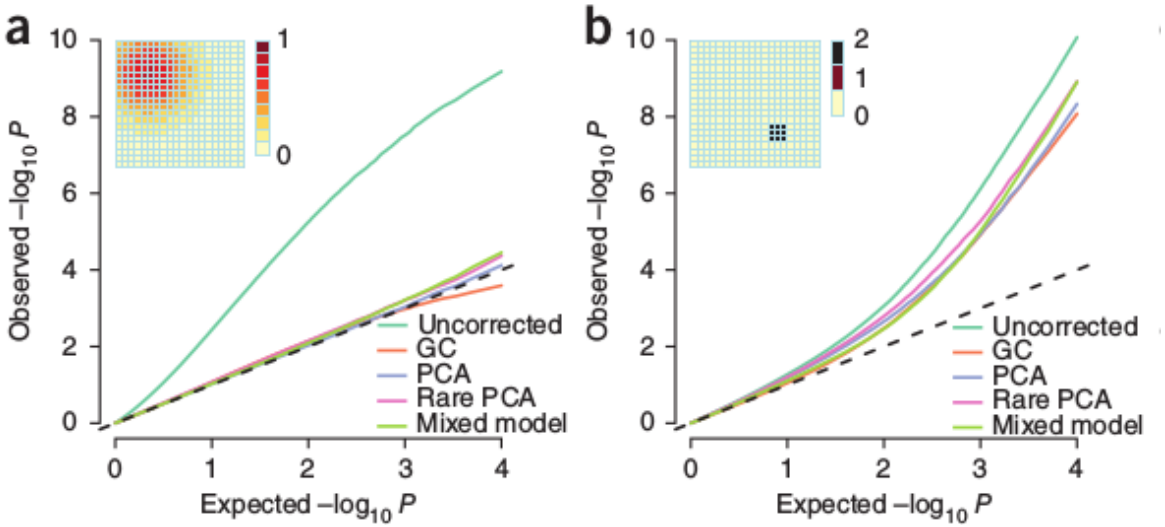
Rare allele association inflation

Differential confounding by allele frequency

The magnitude of confounding due to stratification is a function of allele frequency and phenotypic distribution

- For a gradual phenotypic distribution.
  - Greater inflation of **common** alleles
- For a sharp phenotypic distribution
  - Greater inflation of **rare** alleles

Rare allele association inflation



Mathieson, Nature Genetics 2012

Existing methods for correction for population stratification do not work for sharp phenotypes (and are particularly ineffective for rare variants).

There are at least 3 problems

1. Common stratification correction methods use linear functions to define risk and do not distinguish a tree-like ancestry particularly well.
    1. Need better estimates of genetic relatedness.
2. Differential genotype/phenotype variances lead to scaling of null test statistic distribution.
    1. Need better estimates of test statistic overdispersion.
3. Finite sample sizes yield overdispersion of association statistic.
    1. Need improvements over use of asymptotic distributions.

Addressing fine-scale stratification

Common stratification correction approach

- Build a variance-covariance matrix between all samples using all variants and identify top axes of variation via PCA. (Eigenstrat)
- Apply correction use the top PCs.

Limitations

- Assumes populations are linearly structured in space.
- Inherently relies on common variants relative to rare variants.

Addressing fine-scale stratification

Typical MATLAB-NCS usage

My MATLAB-NCS experience

- "printf" style model building is limited
  - Easy to make errors
  - Small changes to model can require many changes
- Saving models and then executing them is limiting
  - Prevents strategies like GA search for models
  - Prevents encapsulation of an "experiment as a script"
- There is typically very little code reuse
- MATLAB is not a particularly elegant text processing language
- MATLAB support for general purpose libraries is weak

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Why Python?

- Open source, free
- Widely used and growing, active scientific community
- Competitive array math package and plotting packages
- Clean language design
- Object oriented, dynamically typed, garbage collected, bytecode compiled
- Efficient
- Enforced indentation!

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

netplot 3D module

My Ph.D. research

Could a cortical microcircuit act as a functional unit for the memorization of correlations of spatio-temporal spike trains?

- Layer structured cortical microcircuit model
- Two input cell groups
  - Key
  - Data
- Output cell group
- Spatio-temporal spike trains as fundamental unit of information

Stimulus patterns

My Ph.D. research: genetic model search

My Ph.D. research: genetic model search

My Ph.D. research: automated pattern matching

Phase space transitions in a recurrent network

Dr. Doursat:

"My goal is to explore bistability and other firing modes of the network through a survey of parameter space (especially E/I to E/I connections) and, from there, reveal what difference synaptic augmentation (SA) and/or reciprocal connections can make (changes in phase space landscape; easiness of phase transitions) when they are added to the model."

Setting up the experiment in Brainlab

- Convert plain .in file into a script
- Create experiment script
- Create data analysis/graphing

Tighter integration with NCS

Idea: eliminate NCS parsing module, and most of NCS' biological knowledge

- Simpler code; just a "flat" simulation engine that accepts GCList
- Less redundancy of code and documentation
- NCS would be more easily used with unstructured networks
- Easier to add new features

Short-term enhancements to Brainlab

- Some BRAIN.methods become brainlab functions
- Data loading and report plotting standardized, expanded
- Subdirectories for job execution (at least as an option)
- Support for new queueing environments
- Convenience functions for executing multiple jobs
- Automated installation, configuration, testing
- Expanded examples library

Use, so far

- A colleague in our lab used an earlier version of Brainlab
- I have made very extensive use of Brainlab in my own experiments
- A colleague in our lab has recently begun using Brainlab in his experiments
- Another lab is beginning to use NCS, and expressed interest in Brainlab

Acknowledgements

Thanks to:

- My committee
- Dr. Harris, chair
- Dr. Goodman, for the Brain Lab
- Brain Lab colleagues