End User Computing with R under Solvency II

Markus Gesmann

R in Insurance, 15 July 2014

Please stand up

Please stand up

- **Remain standing** if you:

Please stand up

- **Remain standing** if you:
  - You understand Dev/Test/Prod environments.

Please stand up

- **Remain standing** if you:
  - You understand Dev/Test/Prod environments.
  - You use a version control system.

Please stand up

- **Remain standing** if you:
    - You understand Dev/Test/Prod environments.
    - You use a version control system.
    - You know what unit tests are.

Please stand up

- **Remain standing** if you:
  - You understand Dev/Test/Prod environments.
  - You use a version control system.
  - You know what unit tests are.
  - You know the difference between requirements and documentation.

Please stand up

- **Remain standing** if you:
  - You understand Dev/Test/Prod environments.
  - You use a version control system.
  - You know what unit tests are.
  - You know the difference between requirements and documentation.
  - You worry about ownership, usage and change control.

Please stand up

- **Remain standing** if you:
  - You understand Dev/Test/Prod environments.
  - You use a version control system.
  - You know what unit tests are.
  - You know the difference between requirements and documentation.
  - You worry about ownership, usage and change control.
  - You worry about backups and access control.

End User Computing

EUC refers to End User Computing systems that are developed by end users without IT function involvement. Examples include use of Excel, Access, Statistical package R etc.

Source: FSA Solvency II: IMAP data review findings

EUC Risk

Unreliable IT environment, technology or tools can compromise the quality and integrity of the data and its processing within the internal model.

Source: Lloyd’s

EUC Control Objective

To ensure that the quality of data and its processing for use in the internal model is maintained.

Source: Lloyd’s

EUC Expected Control

Controls are established, such as:

- logical access management;
- development and change management (infrastructure, applications, and database);
- security (network and physical);
- business continuity;
- incident management and reporting, and;
- other operational controls that support the collection (including data feeds), storage, analysis and processing.

Source: Lloyd’s

Thus, it’s all about …

- Lineage:
  - Code
  - Data
  - Users
- Quality control:
  - Documentation
  - Testing
- Application dependencies
  - R version, package versions, etc.

This sounds familiar

- CRAN policies:
  - Maintainer
  - Authors
  - Package dependencies
  - Help files
  - Vignettes
  - Unit test
  - Package archives
  - R CMD CHECK pass

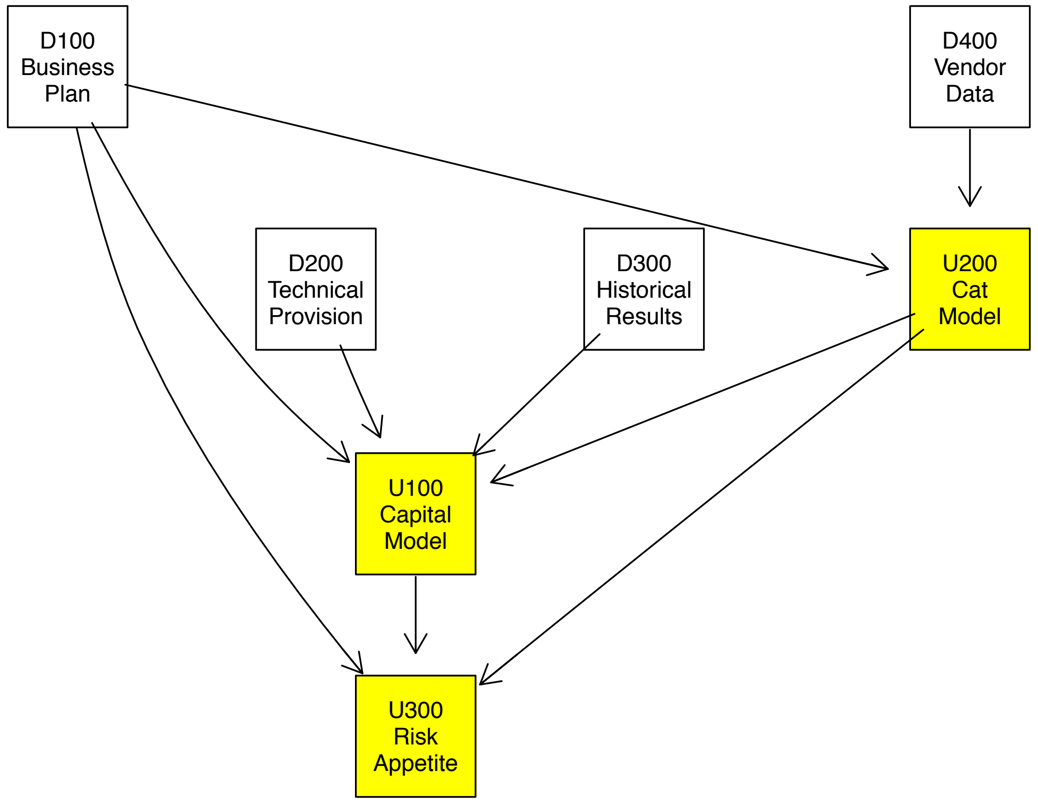
This sounds familiar

- CRAN policies:
  - Maintainer
  - Authors
  - Package dependencies
  - Help files
  - Vignettes
  - Unit test
  - Package archives
  - R CMD CHECK pass

- EUC policies:
  - Owner
  - Authors
  - Lineage
  - Documentation
  - Requirements
  - Testing
  - Version control
  - Automated testing

Example: Data lineage

Data lineage ...



... from data source to usage

Graph created with Rgraphviz from Bioconductor

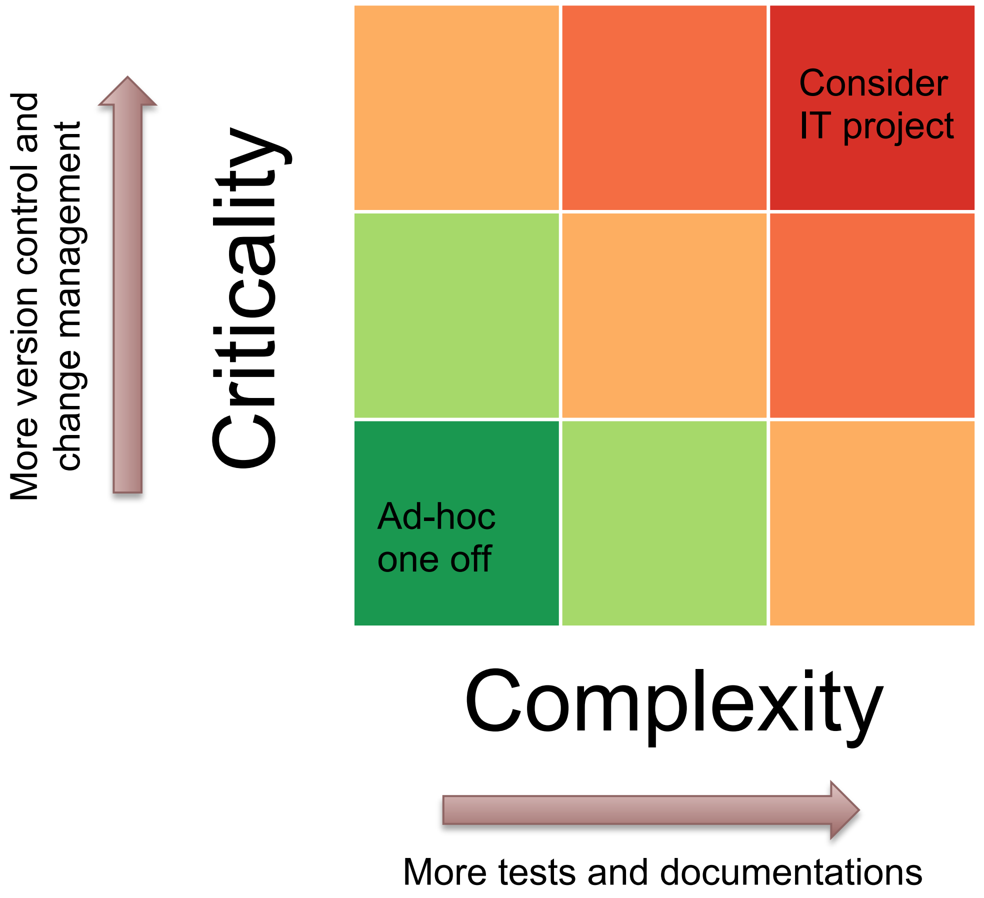
Suggestions

- Follow R manual on writing extensions
- Build packages for your code
  - Document functions with roxygen2
  - Include test
  - Write a vignette for requirements and user documentation
  - Source data from databases
  - Use version control server
  - Set up a local repository for your packages
  - Use virtual machines (docker)

Suggestions

- Ad-hoc work
  - Create R markdown file to collate code and documentation
    - Converted into HTML/PDF/DOCX
  - Version control your code
  - Consider packrat for package management

Summary



More version control and change management
Criticality
Consider IT project
Ad-hoc one off
Complexity
More tests and documentations