Tools for linking Symbolic, Numerical and Statistical Computing Environments.

Richard P. Waterman

Statistics Department

Wharton

U. Penn [a]

September 21 1997

[a] http://www-stat.wharton.upenn.edu/ waterman/homepage.html

Outline

- Motivation – where the problem came from
- Perspective
- Model
- Approach

Motivation

- Colleague wants to apply second order inference techniques
- I want to facilitate this – right thing to do
- Implementation of inference routines requires multiple computing environments
- Turns out not to be very practical for him to implement – since he just wants initially to preview the methods
- **Objective:** make routines available in a straight forward way.

Techniques

◇ Projected score methods – approximate conditional inference.

Take score functions for parameters of interest.

Project onto appropriate subspace to construct a more desirable estimating function.

Desirable in the sense of being closer to the conditional score.

◇ What's involved?

What's involved?

- Space of functions, Λ_∞, generated by centered likelihood ratios
- Basis for $\Lambda_\infty = \{V_i; i = 1, \cdots, \infty\}$
- $V_i = f^{(i)}/f$ **Differentiation.**
- $V_{i+1} = V_i' + V_i V_1$ **Recursion.**
- Projection $\Rightarrow$ inner-product
- $\langle g, h \rangle_\theta = \int g\, h\, dF_\theta(x)$ **Expectation.**
- Symbolic environment is ideal

Modeling and implementation

- Typically happening in a regression type context
- Model specification – factors, contrasts etc.
- Statistical environment is ideal
- Eventually it becomes a large linear algebra problem
- Numerical environment is appropriate for solving the equations
- Symbolic, Statistical and numerical environments.
- Eg, Mathematica, S-Plus and Fortran
- Or, Maple, LispStat and C
- Want to be able to exchange components

Perspective

- Use the right tool for the job – environment that is best suited to the task

    Larger user base

    Continuously developed

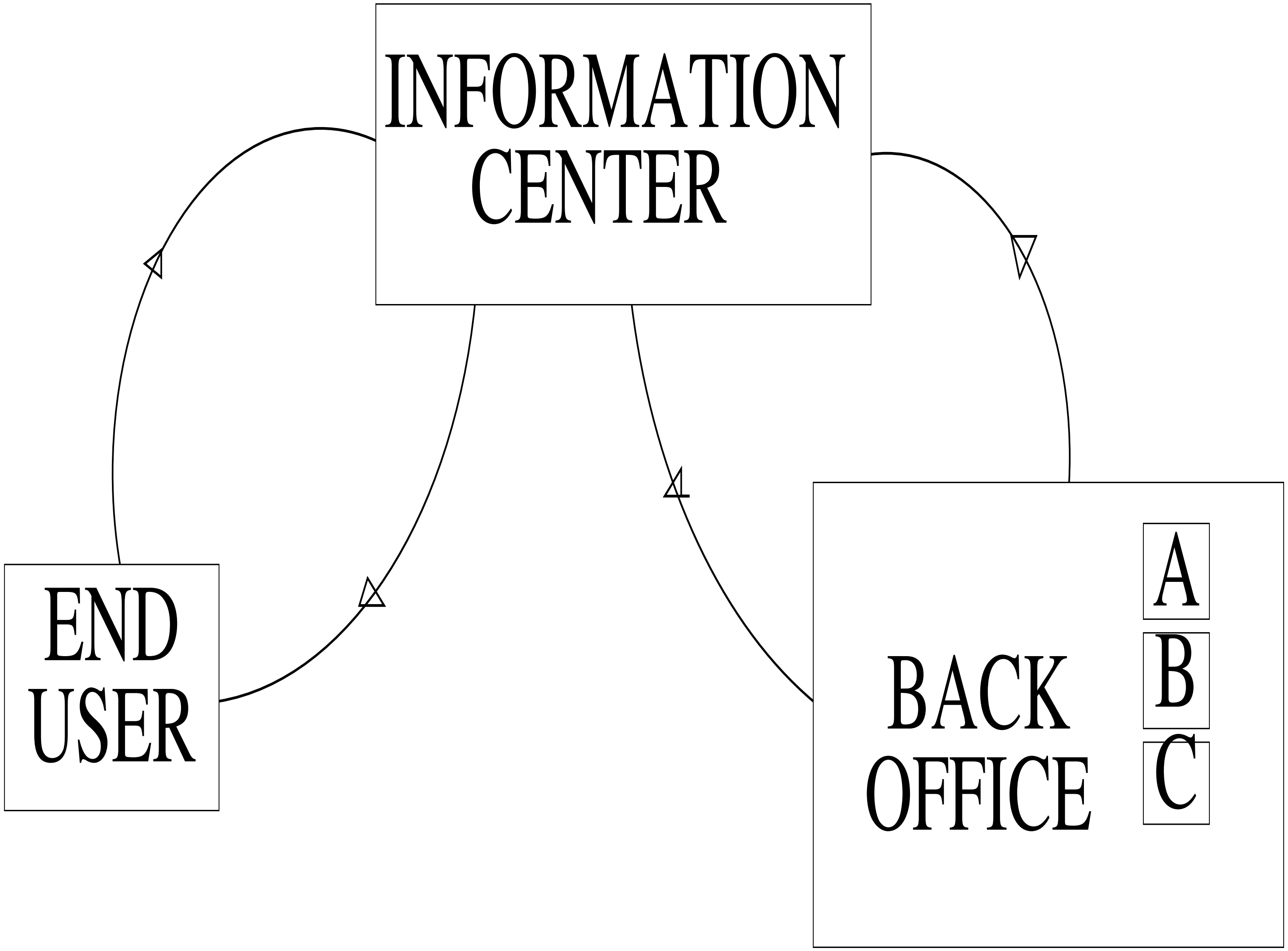
    More reliable

- Really dynamic setting: platforms, programs etc, can not rely on any one component as being stable
- Need to link these components all together
- But provide a reasonable interface to the *ensemble*

Perspective (cont.)

- ◇ HTTP protocol and WEB servers, browsers are ubiquitous
- ◇ Cross platforms – Unix, Windows, Mac
- ◇ Familiarity for most users
- ◇ Almost universal access
- ◇ Only logical to make this the front end

Model



Why split into three?

- Info Center may be sacrificial
- Back office destruction would be more catastrophic
- Allows Info Center to be a “clearing house”
- Back office can be anywhere
- Proprietary data or proprietary code with a trusted intermediary
- Not client/server. All components can deliver information.

Process

$\boxed{1}$ User selects routine from front office menu

May send information as well via a `cookie`

Gets back dynamically created form with embedded help information.

$\boxed{2}$ Fills out form and posts it back with information on:

- Data locations and retrieval method
- Options
- State decisions, delivery options, formatting

Process (cont.)

$\boxed{3}$ Information center processes options

Spawns job to retrieve data

Creates session ID

Saves relevant information in data base

Generates “object” comprising data, processing options, formulae etc.

Sends object to back office

Process (cont.)

$\boxed{4}$ Object arrives at back office

Script fires up – passing elements to appropriate environments

Organizes processes, arranges output into “output object”

Post output object back to information center

$\boxed{5}$ Object arrives back at center

Fires up script, checks session ID, keys into the database, checks delivery options and delivers output to original sender

How to get this done?

Need a tool that has:

- ◇ Good WEB based skills – communication
- ◇ Good database skills – dealing with the information center
- ◇ Good scripting capabilities – for back office processing

ANSWER: $\boxed{\text{PERL}}$

Perl

- A language for easily manipulating text, files and processes
- Interpreted – quick for development and prototyping, and fun
- Derivative language, looks like C, Basic, awk, sed, even C++
- Perl5 – object orientated
- Free & freely available. Lots of code around
- Key modules for these tasks (available from CPAN)

  CGI.pm – Generating forms.

  LWP.pm – Connecting to remote servers etc.

  HTML.pm – HTML creating, parsing and formatting.

Perl (cont.)

Examples – code snippets

- To create a form: `$q = new CGI;`
- To parse a cookie: `%choices = cookie('choices');`
- To create an agent: `$agent = new LWP::UserAgent;`
- To fetch the data:

```
$request = new HTTP::Request('GET',$url);
$response = $agent->request($request);
```

Code snippets (cont.)

```
TR(
td(``Link function''),
td(popup_menu( -name    => LINK_FUNCTION,
               -values  => ['canonical','logit','log','inverse'],
               -default => $choices{LINK_FUNCTION}))),

TR(
td(``Start vals.''),
td(textfield(    -name => START_VALS,
              -default => $choices{START_VALS},
                 -size => 50))),
```

Wrap up and comments

Work

- Creating more extensive classes and objects for passing statistical type information around
- Doing useful things with the objects when they come back, e.g. EDA on the residuals
- Benefits: not focused on a single package at the back-end.
- Update the back end code is a simple process – e.g. bug fix, because it is only in one place.
- Based around Perl, free.
- Succeeds in providing a platform independent, familiar interface to a complex set of routines.

References for Perl

- The Perl Journal: `http://tpj.com`
- The Perl Language Home Page: `http://www.perl.com/perl/`