Zelig: Everyone's Statistical Software

Toward A Common Framework for Statistical Analysis & Development

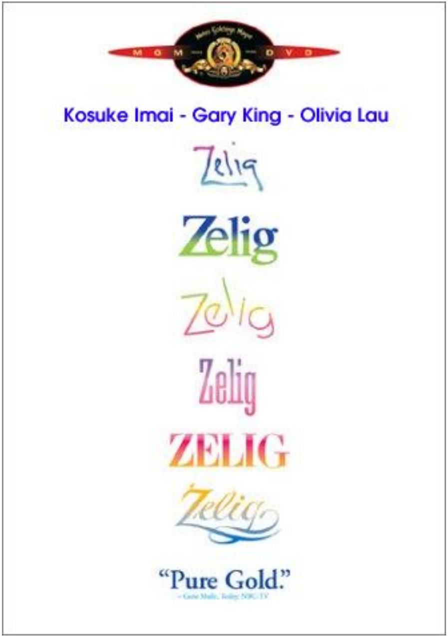
Kosuke Imai[1] Gary King[2] Olivia Lau[3]

[1]Department of Politics
Princeton University

[2]Department of Government
Harvard University

[3]Center for Drug Evaluation and Research
Food and Drug Administration

February 26, 2010



Kosuke Imai, Gary King, and Olivia Lau. "Toward A Common Framework for Statistical Analysis and Development" *Journal of Computational and Graphical Statistics*, Vol. 17, No. 4 (December), pp. 892-913

Motivation for the Zelig Project

- The Problem:
  - Quantitative methodology is thriving like never before
  - It is wonderful, but results in different jargon, notation, syntaxes, etc.
- The Consequence:
  - Hard to learn useful methods developed in various disciplines
  - Despite their common underlying statistical foundation
- Possible Solutions:
  - Top-down approaches: possible (though inefficient) in commercial packages but contradict with the nature of scientific inquiry
  - Open source approaches such as R (and Zelig)

What's R? Why Should I Use R and Zelig?

- What's R?
  - Canned statistical packages
  - An open-source project (free *and* reliable)
  - An object-oriented programming language
- Why should I use R?
  - Most methodologists and statisticians use R
  - More statistical procedures than other software
- Why should I use Zelig?
  - With Zelig, R is *easy* to learn and use
  - No need to wait until a commercial statistical package decides to include a procedure
  - R and Zelig create a community of users and developers of statistics with a common language

What Does Zelig Do?: A Unified User Interface

- Interpreting and presenting statistical results:
  1. Focus on the scientific quantities of interest
  2. Point and uncertainty estimates
- Providing additional infrastructure:
  1. multiple imputation
  2. matching methods
  3. counterfactual evaluations
  4. replication, etc.
- Encompassing a large fraction of statistical models:
  1. Bayesian and frequentist models
  2. single and multiple equations models
  3. cross-section and time-series models
  4. time-series-cross-section models
  5. single and multi-level models, etc.

What Does Zelig Do?: A Developer's Interface

1. **Tools for writing new models** so that developers can easily transform user inputs into mathematically convenient forms
2. **Methods to wrap existing packages** so that developers do not have to modify their packages in order to include them into Zelig
3. **A dynamically-generated GUI** so that those who do not know R can easily use developer's packages

Primary Zelig Commands: An Example

`z.out <- zelig(vote ~ race + educate,`	Select vars
`data = turnout,`	Select data set
`model = "probit")`	Select model
`x.out <- setx(z.out, educate = 12)`	Select QIs
`s.out <- sim(z.out, x = x.out)`	Calculate QIs

Statistical Ontology: R Formula and Its Extension

- R formula is simple yet comprehensive:

```
f <- y ~ x1 + x2:x3 + I(x4^2/sqrt(x3)) + log(x5)
```

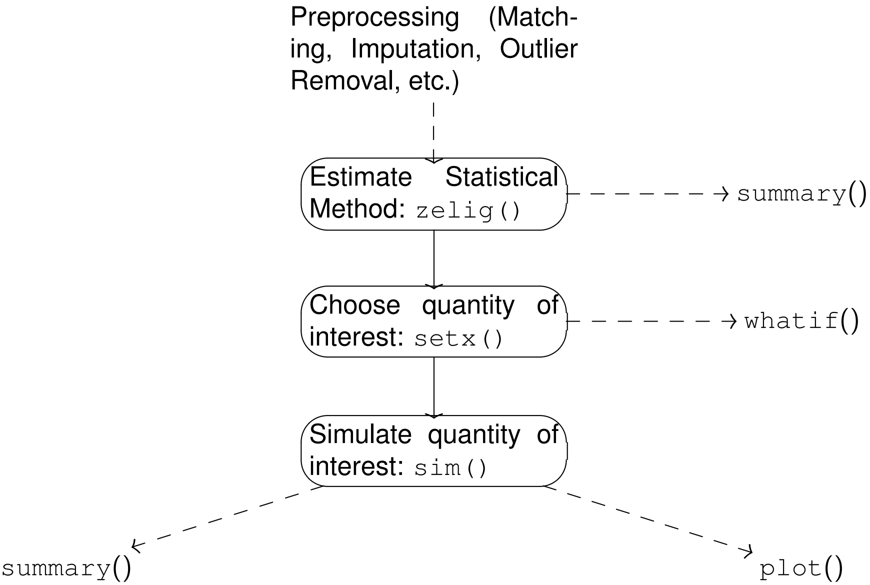
- Zelig's extension of R formula to multiple equations:

```
f <- list(mu1 = y1 ~ x1 + x2 + x3,
          mu2 = y2 ~ x1 + x4 + x5)

f <- list(mu1 = y1 ~ x1 + tag(x2, beta2),
          mu2 = y2 ~ x3 + tag(x4, beta2),
          rho = ~ z1 - 1)

f <- list(cbind(y1, y2) ~ x1 + x2)
```

Built-in Zelig functionality

- Handle multiply-imputed data frames for missing data problems
- Stratifies data and fits a statistical model within each strata
- Works with **MatchIt** which implements a variety of matching methods to reduce model dependence for causal inference
- Works with **WhatIf** which evaluates the validity of counterfactual questions
- Computes various quantities of interest and uncertainties via simulation (bootstrap or Bayesian posterior simulation)
- Numerically and graphically summarizes the results

A Big Picture



Getting Started

- Working directory: `setwd("/Users/kimai/research")`
- Workspace (or global environment)
- Store objects in the workspace: `a <- 5`
- Choose intuitive names for your objects
- R is case sensitive! (`Hello` $\neq$ `hello` $\neq$ `HELLO`)
- Get help using `help.zelig()`

Different Types of R Objects

- Scalar: numbers, character strings, logical values
```
a <- 5; b <- "hi"; c <- TRUE;
```
- Vector: sets of one type of scalar value
```
a <- c(1,2,3); b <- rep(1, 5);
```
- Matrix: 2-D sets of one type of scalar value
```
a <- matrix(c(1,2,3,4), ncol = 2, nrow = 2)
```
- Array: K-D sets of one type of scalar value
```
a <- array(1:30, dim=c(2,3,5))
```
- List: Any combination of the above!
```
obj <- list("first" = a, "second" = b)
```
- Data frame: A special list containing variables of different types

Helpful Functions

- Display objects in the workspace:

```
> ls()
[1] "z.out"   "turnout"
```

- Display elements in an list:

```
> names(turnout)
[1] "race"  "age"  "educate"  "income"  "vote"
```

- Display the dimensions of a data structure:

```
> dim(turnout)
[1] 2000   5
```

Helpful Operators

- Extract one element of a vector, array, or matrix

```
> turnout[25,]
    race age educate income vote
25 white  47      16  5.233    1
```

- Extract an element from a list

```
> turnout[[4]]
```

- Extract a named element from a list

```
> turnout$race <- as.integer(turnout$race)
```

Loading Data

- Tab- or space- delimited .txt file:

```
white   60    14   3.346     1
...
> mydata <- read.table("data.txt")
```

- Comma-separated value .csv file

```
white,   60,    14,   3.346,     1
...
> mydata <- read.csv("data.csv")
```

- Stata .dta file

```
> library(foreign)
> mydata <- read.dta("data.dta")
```

- SPSS .sav files

```
> library(foreign)
> mydata <- read.spss("data.sav",
                      to.data.frame = TRUE)
```

Data Verification

Check to see if the data loaded correctly

- Basic commands:

```
dim(mydata)
summary(mydata)
```

- Check variable names:

```
names(data)
names(data) <- c("income", "educate", "year")
```

- Display specified observations:

```
mydata[2:8, ]
```

Creating New Variables

1. Insert a new variable

```
mydata$new <- new.var
```

2. Merge two data frames

```
new <- merge(x, y)
new <- merge(x, y, by.x  = "x1",  by.y = "y2")
new <- merge(x, y, all = TRUE)
```

3. Edit your data frame like a spreadsheet

```
turnout <- edit(turnout)
```

(Not recommended, but may be useful for some)

Recoding Variables

1. Extract the variable you would like to recode

```
var <- mydata$var1
```

2. Recode the variable

```
var[var < 0] <- 0
```

3. Return the variable to your data frame

```
mydata$var1 <- var
```

Keep the rows in the same order!

Saving R Objects to Disk

- After cleaning your data, you should save it:
  - As an R data file:

    ```
    save(mydata, file = "mydata.RData")
    ```
  - As a tab-delimited file:

    ```
    write.table(mydata, file = "mydata.tab")
    ```
  - As a stata file:

    ```
    library(foreign)
    write.dta(mydata, file = "mydata.dta", version = 10)
    ```
- Alternatively, save your entire R workspace:

  ```
  save.image(file = "Sept1.RData")
  save(mydata, my.function, file = "mydata.RData")
  ```
- To load your `.RData` files back into R:

  ```
  load("mydata.RData")
  ```

Example 1: Logistic Regression

- Question: *Ceteris paribus*, how does age affect voting behavior among
  - High school graduates (12 years of education)?
  - College graduates (16 years of education)?
- The model:

$$\begin{aligned} Y_i &\sim \text{Bernoulli}(y_i \mid \pi_i), \\ \pi_i &\equiv \Pr(y_i = 1 \mid x_i) = \frac{1}{1 + \exp(-x_i\beta)} \end{aligned}$$

- Estimate the model via `zelig()`:

$$\begin{aligned} x_i\beta = \ & \beta_0 + \beta_1\text{Race} + \beta_2\text{Educate} + \beta_3\text{Age} + \beta_4\text{Age}^2 \\ & + \beta_5\text{Income} \end{aligned}$$

```
z.out <- zelig(vote ~ race + educate + age +
                      I(age^2) + income,
               model = "logit", data = turnout)
```

- Set explanatory variables via `setx()`:

	Intercept	Race	Educate	Age	Age^2	Income
x_{12}	1	1	12	18	324	3.9
	1	1	12	⋮	⋮	3.9
	1	1	12	95	9,025	3.9
x_{16}	1	1	16	18	324	3.9
	1	1	16	⋮	⋮	3.9
	1	1	16	95	9,025	3.9

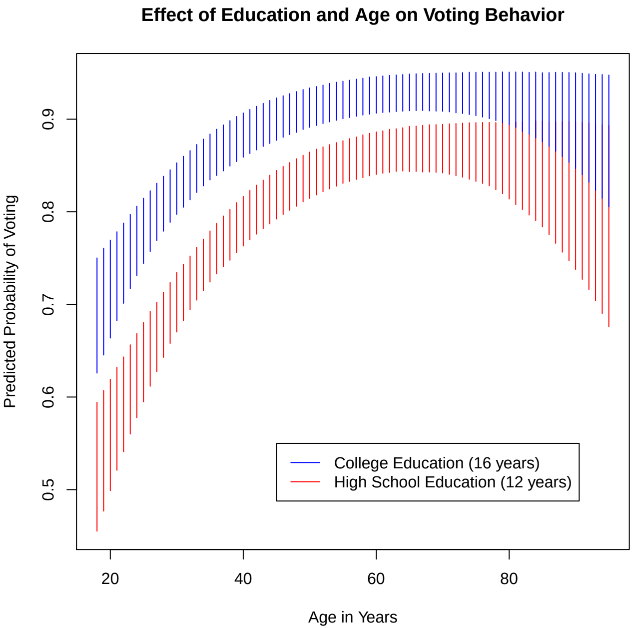
```
x.lo <- setx(z.out, educate = 12, age = 18:95)
x.hi <- setx(z.out, educate = 16, age = 18:95)
```

- Simulate quantities of interest:
  - Simulate $\widetilde{\beta}$ from
    1. asymptotic distribution
    2. Bayesian posterior distribution
    3. sampling distribution using bootstrap
  - Calculate quantities of interest
    1. predicted probabilities: $\widetilde{\pi}_i = 1/(1 + \exp(-x_i \widetilde{\beta}))$ for $i = (12, 16)$
    2. first differences: $\widetilde{\pi}_{16} - \widetilde{\pi}_{12}$
    3. predictive draws: $Y_i \sim Binomial(\widetilde{\pi}_i)$

```
s.out <- sim(z.out, x = x.lo, x1 = x.hi)
```

- Summarize the results:

```
summary(s.out)
plot(s.out)
plot.ci(s.out, xlab = "Age in Years",
        ylab = "Predicted Probability of Voting",
        main = "Effect of Education and Age")
```


Effect of Education and Age on Voting Behavior
Predicted Probability of Voting
0.5
0.6
0.7
0.8
0.9
20
40
60
80
Age in Years
College Education (16 years)
High School Education (12 years)

Example 2: Fixed Effects Models

- Question: Does volume of trade affect unemployment?
  - controlling for overall health of the economy (GDP)
  - controlling for degree of exposure to trade shocks (CapMob)
- Estimate the model:

```
z.out <- zelig(unem ~ gdp + trade + capmob +
                      as.factor(country),
                model = "ls", data = macro)
```

- Set explanatory variables:

	Intercept	GDP	Trade	CapMob	US	Japan
x_{US}	1	3.25	57.08	-0.89	1	0
x_{Japan}	1	3.25	57.08	-0.89	0	1

```
x.US <- setx(z.out, country = "United States")
x.Japan <- setx(z.out, country = "Japan")
```

- Simulate quantities of interest:

```
s.out <- sim(z.out, x = x.US, x1 = x.Japan)
summary(s.out)
```

- *Ceteris paribus*, unemployment is lower in Japan than in the United States:

	Mean	SD	2.5%	97.5%
$E(Y \mid X_{US})$	11.37	0.65	10.14	12.67
$E(Y \mid X_{Japan}) - E(Y \mid X_{US})$	−4.63	0.55	−5.67	−3.54

Example 3: Model Fitting in Strata

Let `data` be a data set with variables `vote`, `age`, `race`, and `state`
To run a model on each state:

1. By hand : a batch file (containing all 50 commands):

```
AL.data <- subset(data, state == "Alabama")
AL <- zelig(vote ~ age + race, data = AL.data,
            model = "logit")
AZ.data <- subset(data, state == "Arizona")
 ...
```

2. A loop:

```
state.name <- order(unique(data$state))
results <- list()
for (i in 1:length(state.name)) {
  tmp <- subset(data, state = state.name[i])
  results[[i]] <- zelig(vote ~ age + race,
      data = tmp, model = "ls")
}
```

With Zelig:

```
z.out <- zelig(vote ~ age + race, data = data,
               by = "state")
```

Example 4: Multiply Imputed Data Sets

- Many data sets come with missing values
- Listwise deletion assumes “missing completely at random”
- Multiple imputation: multiply impute missing values based on the prediction of a statistical model while accounting for the uncertainty about the imputation
- Zelig syntax for the ordinal logit model:

```
z.out <- zelig(as.factor(ipip) ~ wage1992 +
               prtyid + ideol, model = "ologit",
    data = mi(immi1, immi2, immi3, immi4, immi5))
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

- `setx()`, `sim()`, `summary()` do their jobs using all multiply imputed data sets., i.e., no syntax change

Example 5: Matching for Causal Inference

- Matching as nonparametric preprocessing for reducing model dependence in causal inference (Ho, Imai, King, & Stuart, 2007)
- The basic idea: making the treatment and control groups similar to each other in terms of pre-treatment covariates
- Question: Do job training programs affect an individual's real earnings?
- **MatchIt** implements a variety of matching techniques:

```
m.out <- matchit(treat ~ age + educ + black +
                         hispan + nodegree +
                         married + re74 + re75,
                 method = "nearest",
                 data = lalonde)
```

- After matching, fit the model you would have fitted without matching anyway

```
z.out <- zelig(re78 ~ treat + age + educ + black +
                   hispan + nodegree + married +
                   re74 + re75 + distance,
           data = match.data(m.out1), model = "ls")
```

  where `distance` is the estimated propensity score

- Computation of the average treatment effect for the treated:

```
x.out0 <- setx(z.out, fn = NULL, treat = 0,
               data = match.data(m.out, "treat"))
x.out1 <- setx(z.out, fn = NULL,
               data = match.data(m.out, "treat"))
s.out <- sim(z.out, x = x.out0, x1 = x.out1)
summary(s.out)
```

Example 3: Multiple Equations Models

- Question: Are import sanctions and export sanctions likely to occur in the same state?
- Bivariate probit model:
  1. Observation mechanism:

$$Y_j = \begin{cases} 1 & \text{if } Y_j^* \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

  2. Latent (unobserved) variable:

$$\begin{pmatrix} Y_1^* \\ Y_2^* \end{pmatrix} \sim N_2 \left\{ \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right\},$$

  where μ_j is a mean for Y_j^* and ρ is a scalar correlation parameter given by,

$$\mu_j = x_j \beta_j \quad \text{for} \quad j = 1, 2,$$

$$\rho = \frac{\exp(x_3 \beta_3) - 1}{\exp(x_3 \beta_3) + 1}.$$

- Default: estimate only the two conditional mean equations with the same set of X and have no X for the correlation parameter

```
z.out <- zelig(cbind(import, export) ~ coop +
               cost + target, model = "bprobit",
               data = sanction)
```

```
x.lo <- setx(z.out, cost = 1)
x.hi <- setx(z.out, cost = 4)
```

```
s.out <- sim(z.out, x = x.lo, x1 = x.hi)
```

- It's possible to specify different variables in each equation:

```
z.out <- zelig(list(mu1 = import ~ coop,
                    mu2 = export ~ cost + target),
               model = "bprobit", data = sanction)
```

- With Zelig, it is even easy to constrain the parameters across different equations:

```
z.out <-
   zelig(list(mu1 = import ~ tag(coop,"coop") +
         tag(cost, "cost") + tag(target, "target"),
              mu2 = export ~ tag(coop,"coop") +
         tag(cost, "cost") + tag (target, "target")),
         model = "bprobit", data = sanction)
```

- `setx()` and `sim()` steps are identical

Concluding Remarks

- Zelig provides a unified interface for both users and developers
- Makes R and its numerous functionalities accessible to applied researchers
- Many more improvements planned for Zelig
  1. Adding more models
  2. Collaboration with the Dataverse Network
  3. API to encourage more contributions
- Visit Zelig on the web at

http://gking.harvard.edu/zelig/