Inference after model selection

The Alan Turing Institute

Joshua Loftus (jloftus@turing.ac.uk)

November 8, 2016

Setting: regression model selection

Linear model

$$y = X\beta + \epsilon$$

- y vector of outcomes
- X predictor/feature matrix
- β parameters/weights to be estimated, assume most are “null,” i.e. equal 0 (sparsity)
- ϵ random errors, assume probability distribution $N(0, \sigma^2 I)$
- Pick subset of predictors we think are non-null
- How good is the model using this subset?
- Are chosen predictors actually non-null, i.e. significant?

Type 1 error: declaring a predictor significant when it is actually null.

Motivating example: forward stepwise

Data: California county health data...
Outcome: log-years of potential life lost.
Model: 4 out of 31 predictors chosen by FS with BIC.

```r
model <- step(lm(y ~ ., df), k = log(n), trace = 0)
print(summary(model)$coefficients, digits = 2)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -0.079       0.13   -0.62   0.5394
## Poverty12mo      -0.310       0.13   -2.35   0.0231
## MedianIncome     -0.393       0.12   -3.34   0.0017
## `%Obese`          0.267       0.12    2.26   0.0288
## InjuryRate        0.284       0.12    2.38   0.0218
```

4 interesting effects, all significant. Time to publish!

What's wrong with this?

What's wrong with this?

The data was actually random: uncorrelated noise!

```r
set.seed(1)
n <- 50
p <- 31
df <- data.frame(matrix(rnorm(n*p), nrow=n))
df$y <- rnorm(n)
names(df)[c(6, 14, 20, 24)] <-
  c("Poverty12mo", "MedianIncome",
    "%Obese", "InjuryRate")
```

(With apologies for deceiving you, I hope this makes the point. . .)

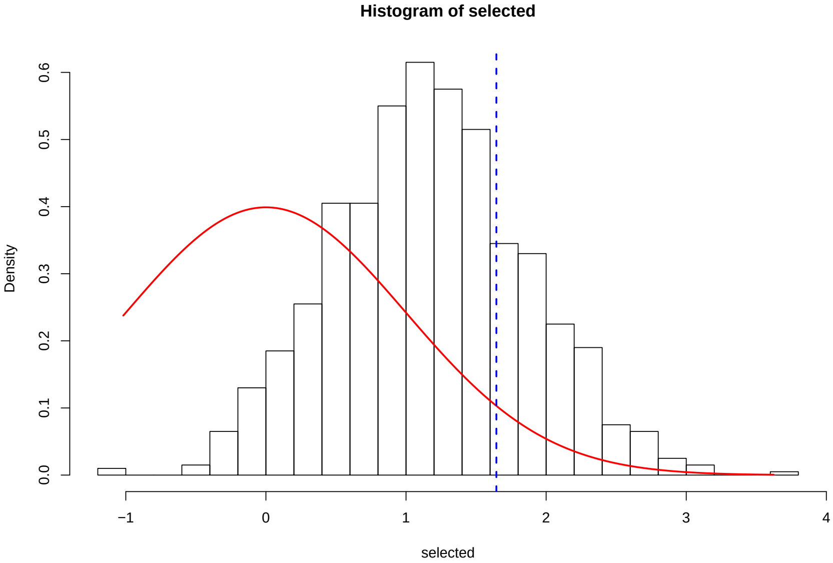
What is going wrong?

Simple example: choose the largest of 5 effects.

```r
maxz <- function(n) return(max(rnorm(n)))
selected <- replicate(1000, maxz(5))
range <- seq(min(selected),
             max(selected),
             length.out = 1000)
```

This data is generated under a global null, all effects have 0 mean. What happens if we now compute p-values for the selected effects using standard normal tail areas?

Selection bias: type 1 error 0.236 instead of 0.05



Selection can make noise look like signal

Any time we use the data to make a decision (e.g. pick one model instead of some others), we introduce a selection effect (bias).

Selection can make noise look like signal

Any time we use the data to make a decision (e.g. pick one model instead of some others), we introduce a selection effect (bias).

Forward stepwise, Lasso, elastic net with cross-validation, etc, all use the data in a way that would result in such bias.

Selection can make noise look like signal

Any time we use the data to make a decision (e.g. pick one model instead of some others), we introduce a selection effect (bias).

Forward stepwise, Lasso, elastic net with cross-validation, etc, all use the data in a way that would result in such bias.

Significance tests, prediction error, R^2, goodness of fit tests, etc, will all suffer from selection bias

Big contributor to reproducibility crisis

> ***We conducted replications*** *of 100 experimental and correlational studies published in three psychology journals using high-powered designs and original materials when available. . . . Thirty-six percent of replications had significant results; 47% of original effect sizes were in the 95% confidence interval of the replication effect size;* ***39% of effects were subjectively rated to have replicated the original result***

From *Estimating the reproducibility of psychological science* (Open Science Collaboration, 2015). See also *Why most published research findings are false* (Ioannidis, 2005).

What's the most common solution?

What's the most common solution?

Data splitting

Before doing any selection, set aside some **validation data**. Then, *after* the final model is chosen, use this validation set to compute prediction error, significance tests, etc.

Survival example: Cox's PH model, regularized

- Data: 240 lymphoma patients, 7399 genes

```
train <- sample(nrow(x), 140)
x.train <- x[train,]
y.train <- Surv(y[train], status[train])
fit <- glmnet(x.train, y.train, family = "cox")
cv.fit <- cv.glmnet(x.train, y.train,
                    family = "cox")
coefs <- coef(fit, s = cv.fit$lambda.min)
active <- which(coefs != 0)
length(active)
```

```
## [1] 15
```

Inference for the selected model

```
test <- setdiff(1:nrow(x), train)
x.test <- x[test, active]
y.test <- Surv(y[test], status[test])
fit.test <- coxph(y.test ~ x.test)
fit.test
```

```
## Call:
## coxph(formula = y.test ~ x.test)
##
##              coef exp(coef) se(coef)     z     p
## x.test1  -0.2730    0.7611   0.2096 -1.30 0.193
## x.test2   0.6954    2.0045   0.4394  1.58 0.114
## x.test3   0.1218    1.1295   0.3748  0.32 0.745
## x.test4  -0.0145    0.9856   0.3038 -0.05 0.962
## x.test5   0.0755    1.0784   0.1918  0.39 0.694
## x.test6  -0.1430    0.8668   0.0648 -2.21 0.027
```

Data splitting

Pros:

- Simple: only took a few lines of code
- Robust: requires few assumptions
- Controls type 1 error, no selection bias

Data splitting

Pros:

- Simple: only took a few lines of code
- Robust: requires few assumptions
- Controls type 1 error, no selection bias

Cons:

- Reproducibility issues: different random splits, different split proportions
- Efficiency: using less data for model selection, also less power
- Feasibility: categorical variables with rare levels (e.g. rare variants)

Selective error control

New and active research area; Taylor, Tibshirani, Fithian, many others.
To adjust for the selection effect, *condition* on the selected model.
Mathematically, if we select M, want test a null hypothesis H_0 about M (e.g. significance test for a variable in M), we want tests that control

Selective type 1 error

$P_{M,H_0}(\text{reject } H_0 | M \text{ selected}) \leq \alpha$

Selective error control

New and active research area; Taylor, Tibshirani, Fithian, many others. To adjust for the selection effect, *condition* on the selected model. Mathematically, if we select M, want test a null hypothesis H_0 about M (e.g. significance test for a variable in M), we want tests that control

Selective type 1 error

$P_{M,H_0}(\text{reject } H_0 | M \text{ selected}) \leq \alpha$

If a variable "surprises" us enough to be *included in the model*, it must surprise us *again* in order to be *declared significant*

- Data splitting controls this error trivially
- Controlling this would fix reproducibility problems

A simple example: marginal screening rule

Observe many (independent) means, select the effects which might be large, say > 1

```r
Zs <- rnorm(10000)
screen <- Zs > 1
Zscreened <- Zs[screen]
mean(screen)
```

```
## [1] 0.1637
```

A simple example: marginal screening rule

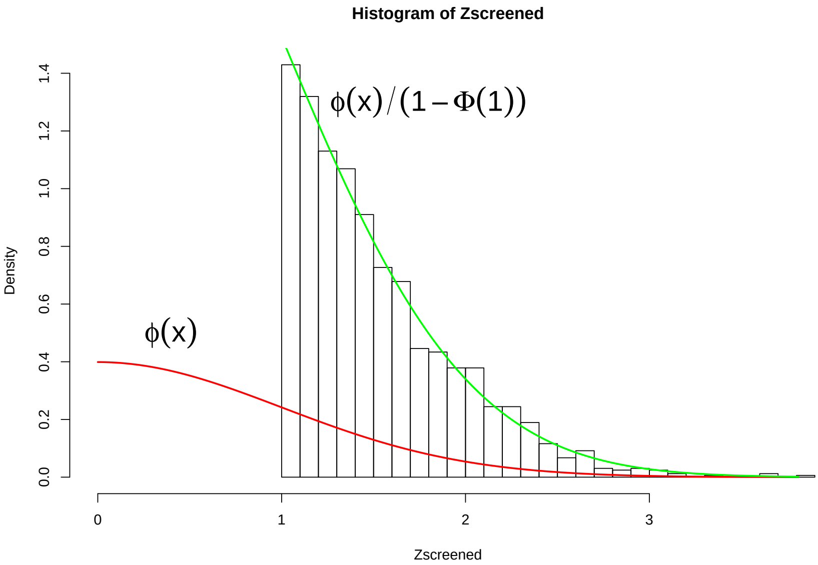
Observe many (independent) means, select the effects which might be large, say > 1

```r
Zs <- rnorm(10000)
screen <- Zs > 1
Zscreened <- Zs[screen]
mean(screen)
```

```
## [1] 0.1637
```

These are all null. What distribution can we compare them to, and still control type 1 error?

Truncated probability law



Details vary depending on procedure

Most of the work in this field is in understanding the geometry of truncation for each particular selection procedure

My work focuses on procedures with complicated geometry (quadratic constraints), and includes a few important special cases (groups of variables, cross-validation)

groupfs simulation: $n = 100$, $p = 100$, $P = 50$

Model size chosen with BIC. Groups 1-4 have $\|\beta\|_2 = .84$ within groups, all else 0

```
> set.seed(1)
  ...
> fit <- groupfs(x, y, ...)
> pvals <- groupfsInf(fit)
> pvals
  Group Pvalue     TF df   Size Ints    Min     Max
1     3  0.088 49.913  2 67.811    1 44.949 112.760
2     1  0.000 98.077  1 54.267    1 68.151 122.418
3     2  0.003 69.266  1 28.659    1 50.423  79.082
4     4  0.000 37.099  2 28.803    1 20.194  48.997
5    47  0.319  5.143  1  3.887    1  3.518   7.406
```

Ignoring selection, first 4 p-values are $<$0.001, for 47 it's 0.024

Remarks

Technical details in the papers, but beware:

- Tests not independent (with one notable exception)
- Some examples are computationally expensive (cross-validation)
- May be low powered against some alternatives

Software implementation: `selectiveInference` R package on CRAN
Github repo: `https://github.com/selective-inference/`

Which method to use for a given problem?

- If n is very large, might just use data splitting (simple)
- Otherwise, consider the conditional approach, especially if $p > n$ or bottlenecks like rare observations limit effective sample size
- If p is small, more robust/conservative method ("PoSI") is available, see Berk et. al. (2013).

Most general takeaway message

Selection is a source of uncertainty

Data science pipelines must **address all sources of uncertainty**. Otherwise we might just be fooling ourselves. . .

References

- Taylor, Tibshirani (2015). Statistical learning and selective inference. **PNAS**.
- Benjamini, (2010). Simultaneous and selective inference: current successes and future challenges. Biometrical Journal.
- Berk et al, (2010). Statistical inference after model selection. Journal of Quantitative Criminology.
- Berk et al, (2013). Valid post-selection inference. Annals of Statistics.
- Simon et al, (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. Journal of Statistical Software.
- Loftus, (2015). Selective inference after cross-validation. arXiv Preprint.
- Loftus and Taylor, (2015). Selective inference in regression models with groups of variables. arXiv Preprint.

Thanks for your attention!

Questions?

Please message (`jloftus@turing.ac.uk`) or find and talk to me anytime.