Replicability II

Data Science 101 Team

Selection and reproducibility

- We test a lot of hypotheses, but we only report the "significant" results.
- This is of course all right – we do not want to spend time describing relations that did not appear real.
- But we need to be aware of this selection when interpreting the strength of the reported results:
- Sometimes results are selected because they arise from unusual random outcomes. Then the standard methods for evaluating them do not apply.

Selection and Reproducibility

- Situation considered today: We use our data to select some parameters and then form confidence intervals for the selected parameters.
- Just as in multiple testing, we should adjust our intervals or otherwise the inference would be distorted.

Soric warned about the practice of reporting confidence intervals following some selection procedure in 1989, saying:

"In a large number of 95% confidence intervals, 95% of them contain the population parameter [. . .] but it would be wrong to imagine that the same rule also applies to a large number of 95% interesting confidence intervals."

Winner's curse

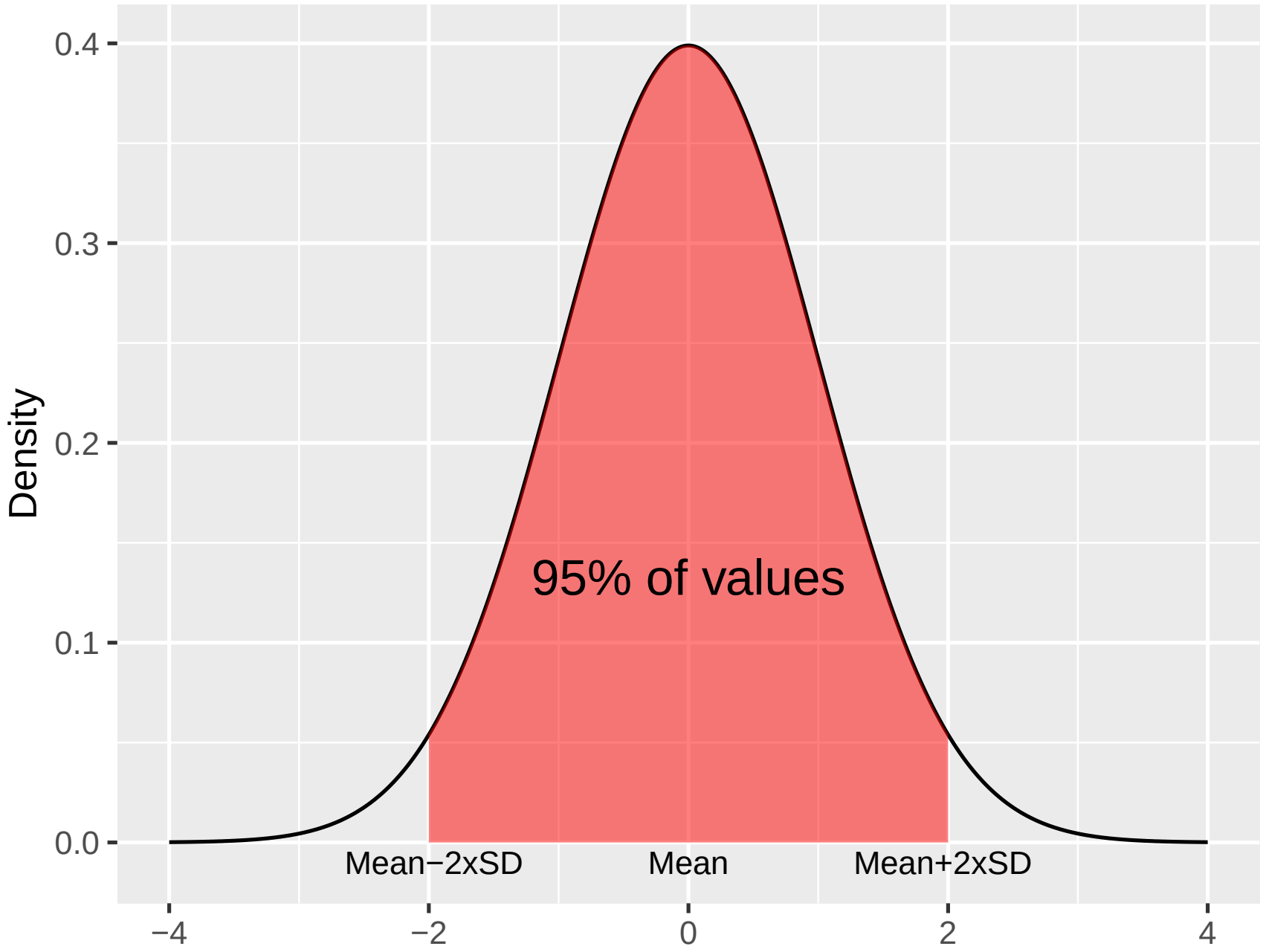
There is a "Winner's curse" in auctions which is different from the winner's curse we will discuss now!

Let's look what happens when we select by "significance"

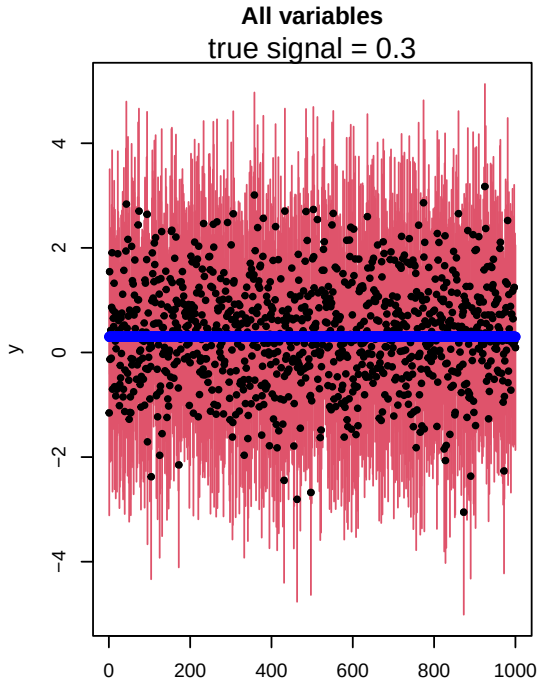
```
mu = 0.3 # mean parameter
N = 1000 # number of obervations
Tmean = rep(mu,N) # all means are equal to 0.3
y = rnorm(1000,Tmean,1)
pvalue = 2*pnorm(-abs(y))
discoveries = pvalue < 0.05
lower = y - 1.96
upper = y + 1.96
```

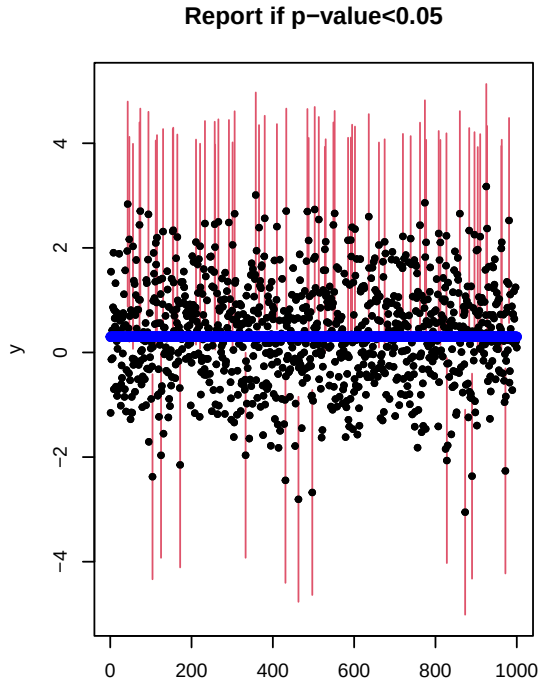
- Note that every hypothesis is non null so that every discovery is true.
- We did not account for multiple testing – this is OK, it just means we will not have the guarantee a multiple testing procedure offers. Think of this as a first stage "screening" process.
- For the "discoveries" we now want to give a good description of the effect size: produce confidence intervals, for example.

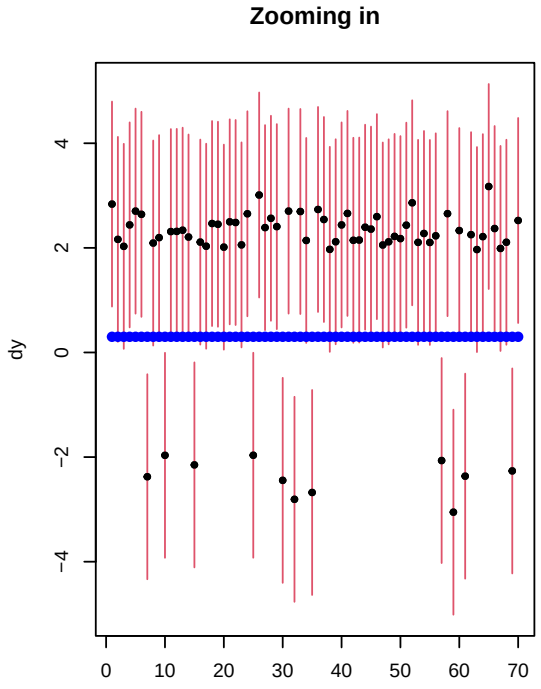
Recall confidence intervals



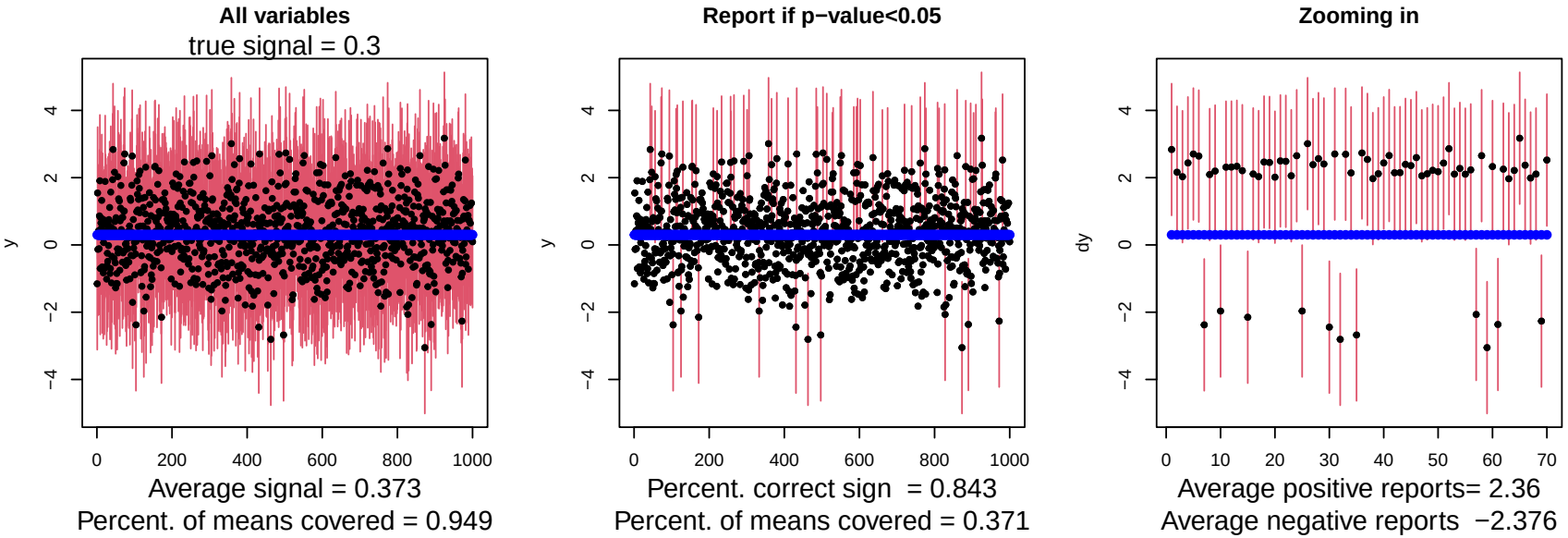
Let's look what happens when we select by "significance"







Let's look what happens when we select by "significance"



- When we look at all variables, we see that the coverage is all right: about 95% contain the true mean 0.3. When we look at the selected variables, this is no longer the case.
- Also, the **sign error is high**: about 20% of estimated effects have the wrong sign.
- **Estimated effects are far bigger than they are (over-estimate effect size)**.

This is related to regression towards the mean

- Example: score = ability + random error. Suppose we focus on the observations with largest (or smallest) values.
- These are going to be the ones with high (low) abilities, but they also have large errors in the same direction. (See bottom two pictures on the following slide.)
- The highest scores pertain to players with both high ability and large random error in their favor.
- This explains why the highest score often does not belong to the player with the highest ability, and on another "tournament" the top scorer typically falls back a bit because the big luck is unlikely to repeat ("winner's curse", "regression to the mean", "regression fallacy").
- Why "fallacy"? There is a temptation to find some cause for the drop (slackening off) when in fact such a drop is to be expected.

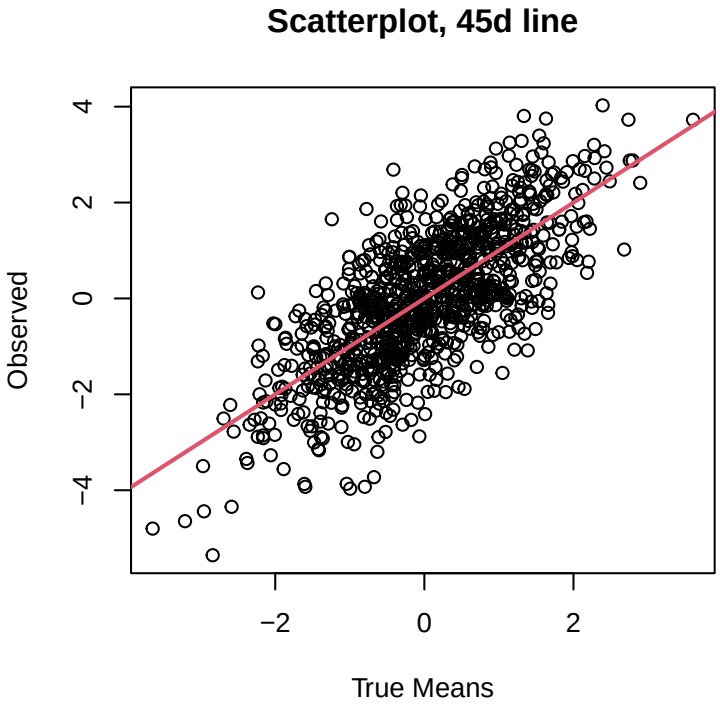
A simulated example

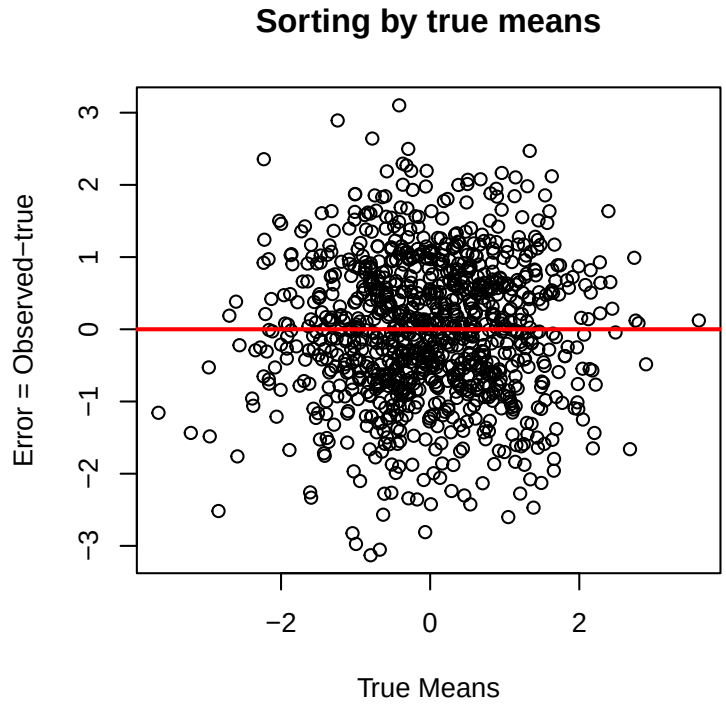
```
M = 1000
Tmean = rnorm(M) # will serve as true mean
y = rnorm(M, Tmean, 1)  # add normal random error to Tmean
```

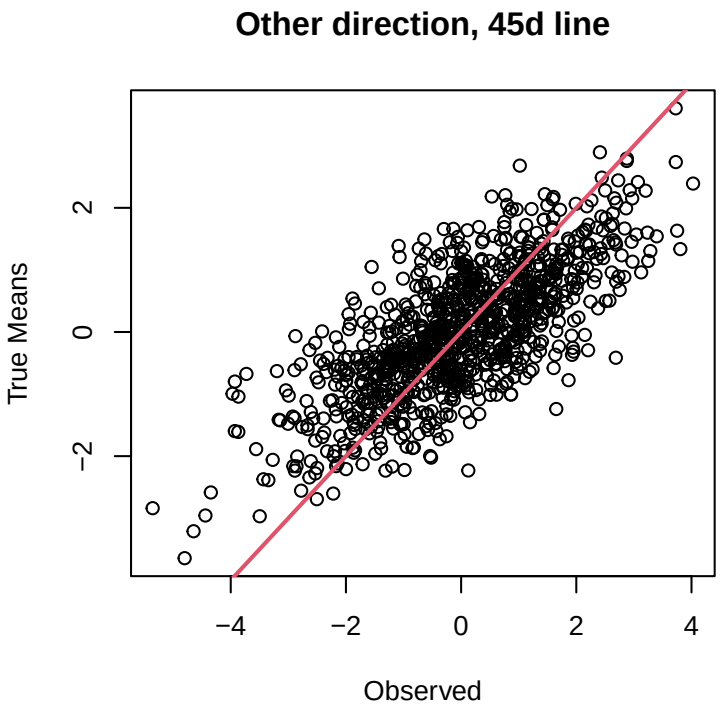
Tmean is the vector of true means ("abilities").

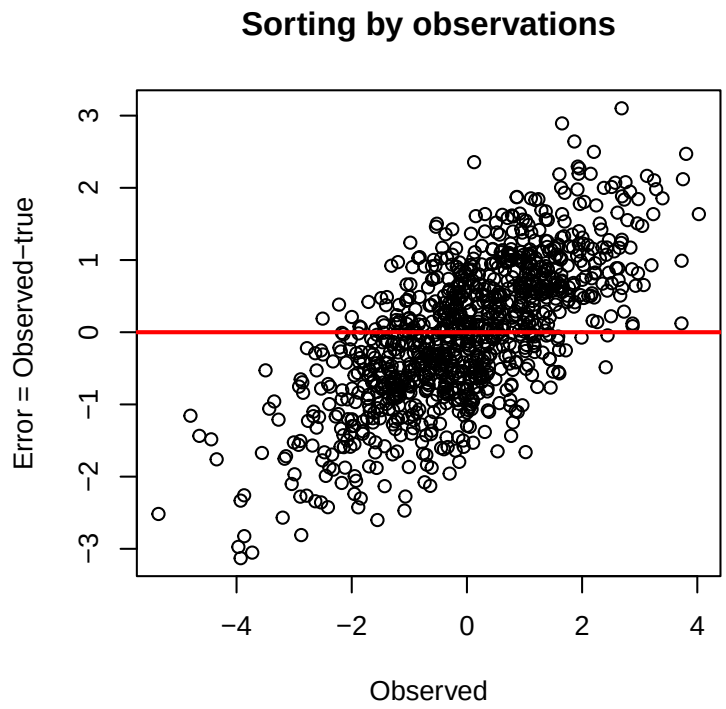
y is the observed outcome ("test score")

y = Tmean + chance error


Scatterplot, 45d line
Observed
True Means
−4
−2
0
2
4
−2
0
2


Sorting by true means
Error = Observed−true
True Means
−3
−2
−1
0
1
2
3
−2
0
2


Other direction, 45d line
True Means
Observed
−2
0
2
−4
−2
0
2
4

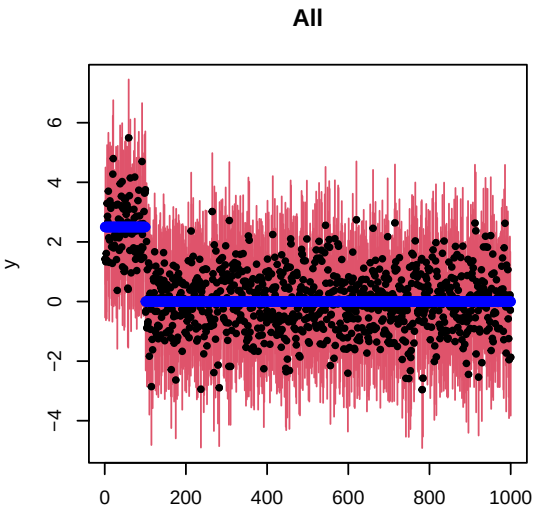

Sorting by observations
Error = Observed−true
Observed
−3
−2
−1
0
1
2
3
−4
−2
0
2
4

Now consider a case where many null hypotheses are tested

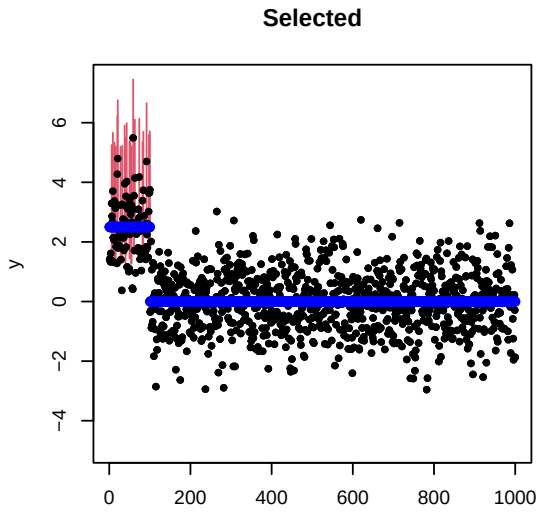
```
Tmean = rep(0,1000) # the majority of signals are zero
N = 100 # number of non-zero means
mu = 2.5 # signal strength
Tmean[1:N] = mu
y = rnorm(1000,Tmean,1)
pvalue = 2*pnorm(-abs(y))
discoveries = p.adjust(pvalue, 'BH') < 0.05
```

- The majority of tested hypotheses is null.
- To identify which hypotheses to reject we use BH.
- We are now accounting for multiplicity, i.e. we have a guarantee about FDR.

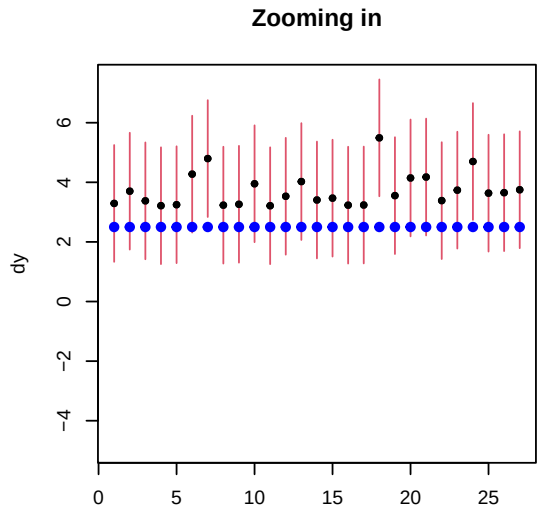
Confidence intervals after BH



Percent. of means covered = 0.944

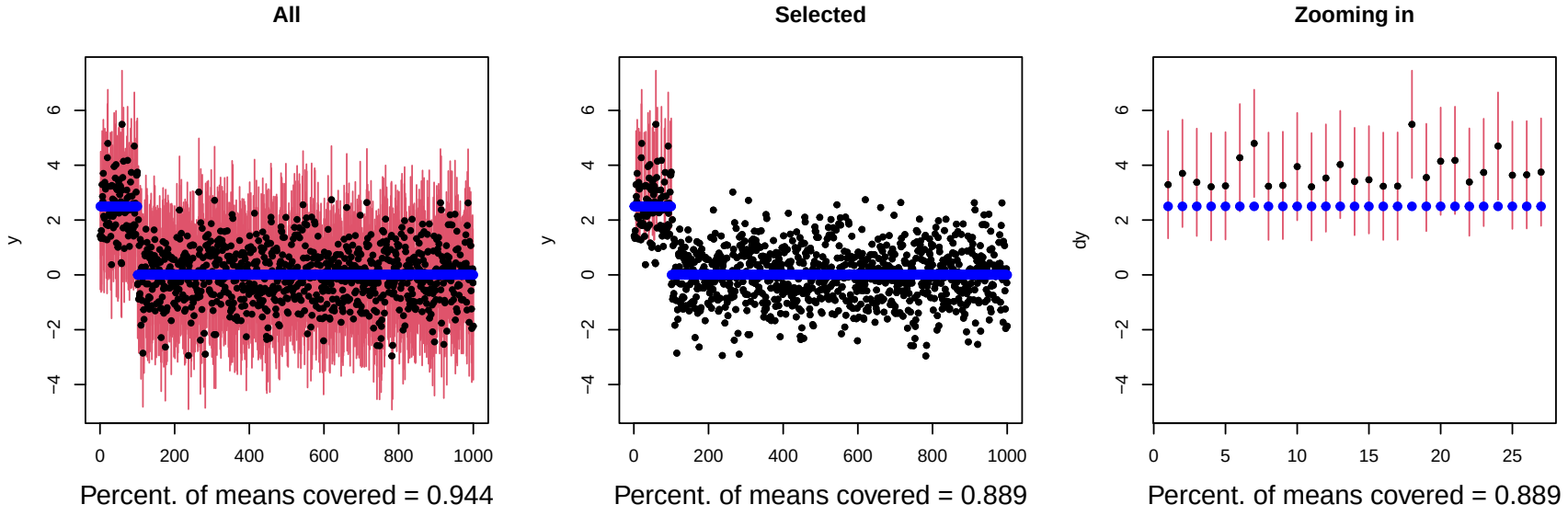


Percent. of means covered = 0.889



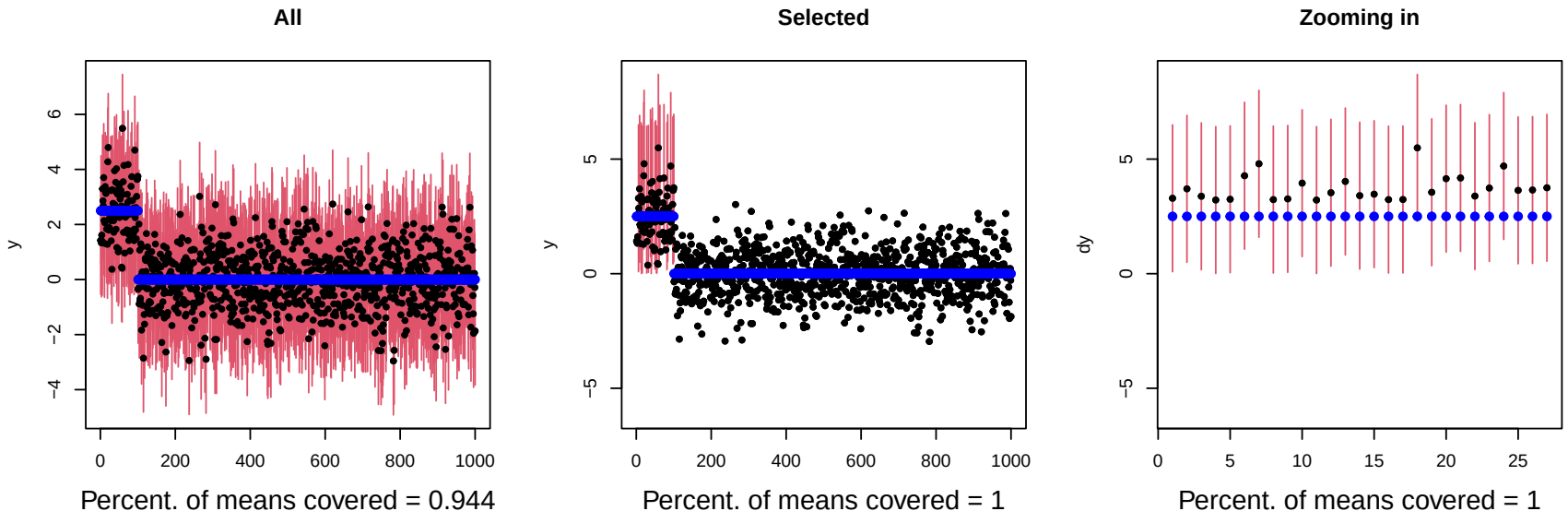
Percent. of means covered = 0.889

Confidence intervals after BH



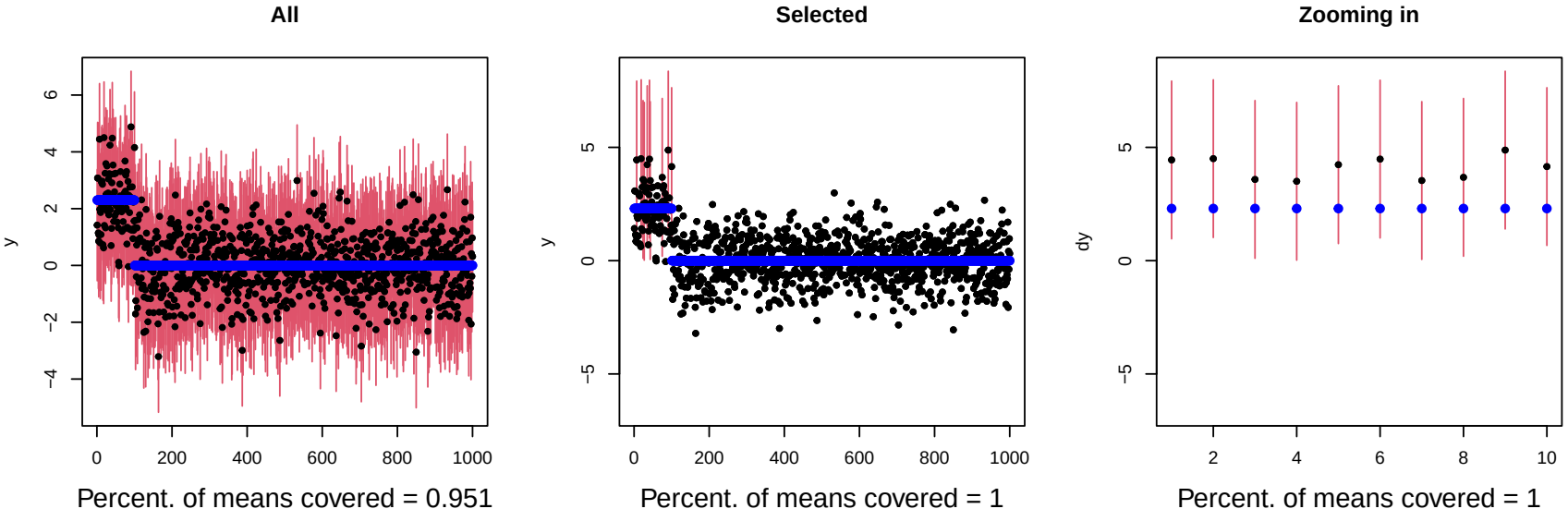
- When we look at all variables, we see that the coverage is all right (by definition).
- When we look at the selected variables, the coverage is better than before but still not correct $\Longrightarrow$ correcting for multiplicity does not correct for winner's curse!
- It is possible to account for this effect and produce different confidence intervals achieving the right coverage - this is a topic of current research.

Confidence intervals after BH



- In this example, a simple fix is to make the intervals wider based on concept of "False Coverage Rate".

Confidence intervals after BH: a simple method to adjust for selection



- If desired coverage level is $1 - \alpha$, then adjust α by dividing by `N / TD`, the ratio of the total number of hypotheses to the number of discoveries discovered by BH.

A different way to deal with selection bias

- A different approach is based on *conditioning* on what we have observed. (No details here.)
- In this approach we record what aspects of the data we have observed: *in silico* this means keeping track of which functions we have applied to our data and what we have observed. Then a certain computation is done conditional on this record.
- A record of this is something a scientist (and / or a *data scientist*) should do in any case.
- As data analysis gets more complicated, so does this record. *Calculations start to become difficult.*

Winner's curse in regression: model selection

- A common way to fit models is to use *model selection*: an algorithm chooses a model that seems *best* by some measure.
- Simple example: We select the X that has the highest correlation with Y.

```
n = 100 # number of cases
p = 10  # number of features
X = scale(matrix(rnorm(n*p), n, p), TRUE, TRUE)
Y = rnorm(n)
# highest correlation
best_var = which.max(abs(t(X) %*% Y))
summary(lm(Y ~ X[,best_var]))
```

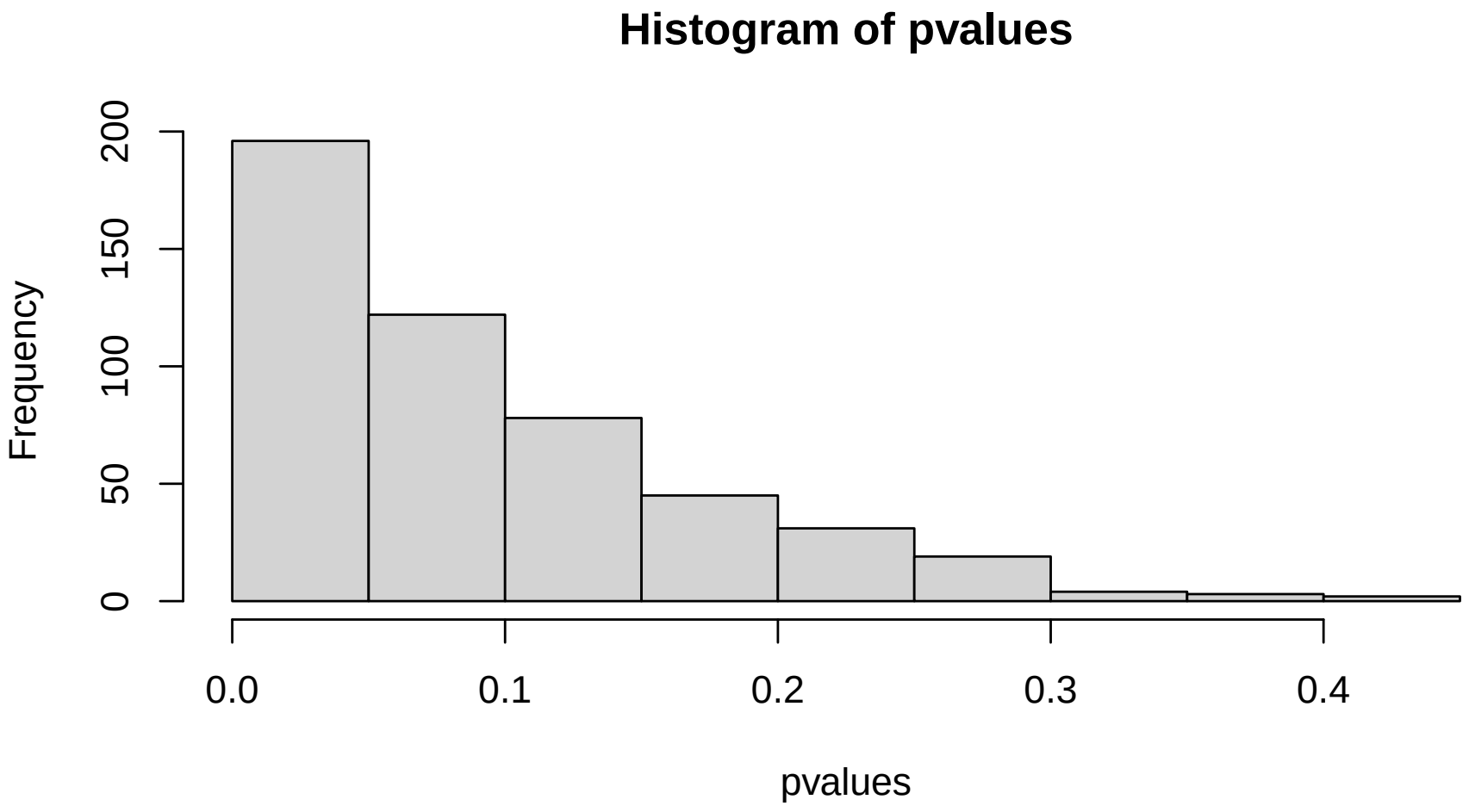
Winner's curse in regression

Wrap into a function and return the p-value for the best variable:

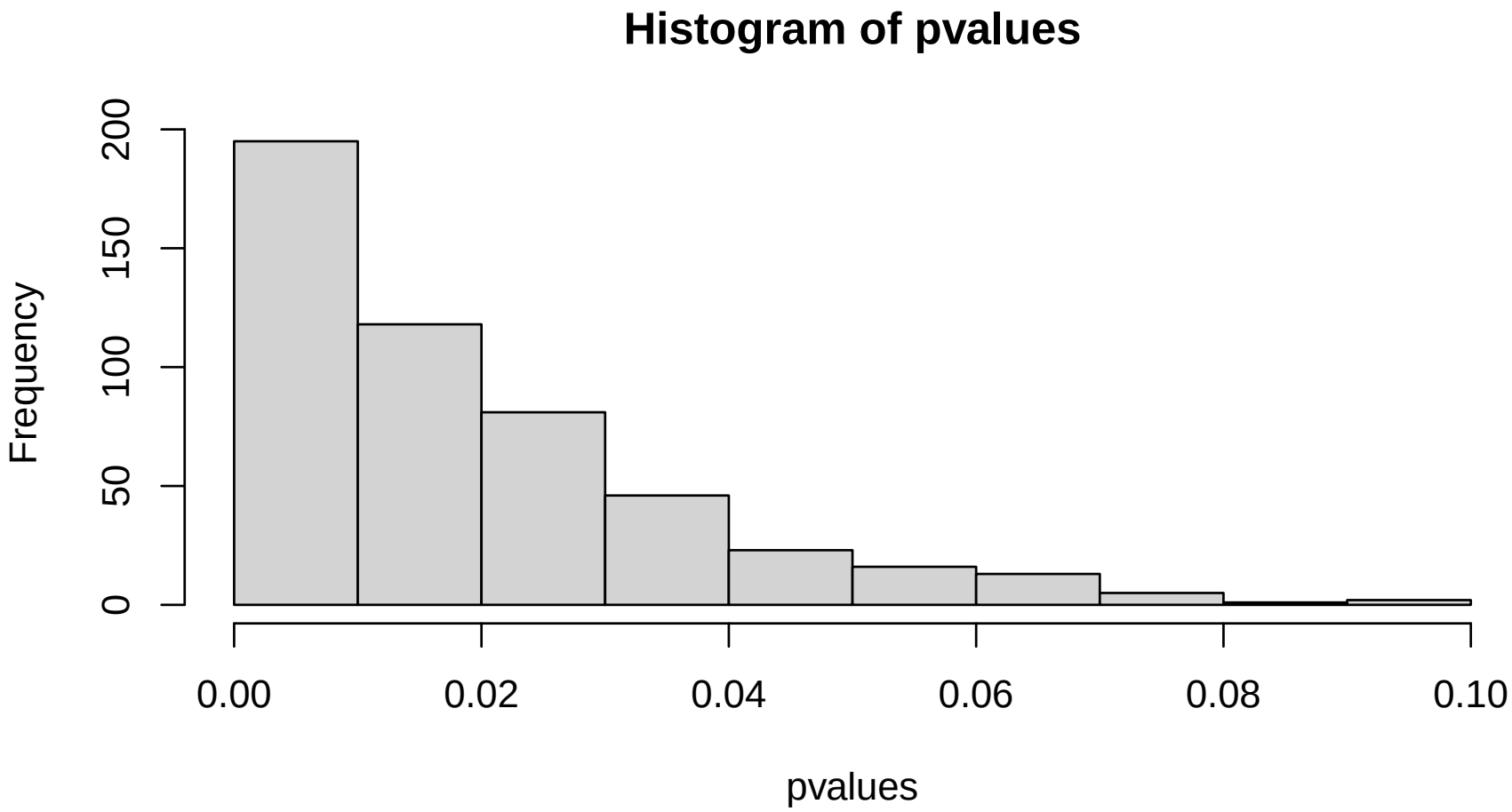
```
best_pvalue = function(n=100, p=10) {
    X = scale(matrix(rnorm(n*p), n, p), TRUE, TRUE)
    Y = rnorm(n) # not related to X!
    # highest correlation
    best_var = which.max(abs(t(X) %*% Y))
    return(summary(lm(Y ~ X[,best_var]))$coef[2,4])
}
```

Histogram for p = 10 features

- Call the previous function e.g. 500 times:



Winner's curse in regression for p=50 features



- Not uniform!
- Thresholding p-value at 0.05 yields a Type I error of 0.93!
- This gets even worse if there are more than 50 explanatory variables!

Reproducible research

- Any scientific experiment needs to be reproducible: another scientist needs to be able to recreate the same result following the same procedure
- Need to precisely keep track of the **procedure**
  - Have some outliers been discarded?
  - Has the data been transformed?
  - Which statistical test has been used to obtain p-values?
  - Which procedure has been used to control for multiplicity?