A Unified View of Model Construction and Evaluation

- We present an effective big data framework for predictive model evaluation.
  - Critical tools for executives, managers, and other data science *consumers.*
- Using this perspective we will work through nested model construction (as in stacking/super-learning).
  - Advanced methods for data science *producers*.

John Mount
Nina Zumel
Win-Vector LLC

Win-Vector LLC

Workshop Outline

- Part 1: Statistics for data science manager and practitioners.
    - Punchline: “statistics for data science” is statistics.
- Part 2: Building reliable nested models using simulated out of sample data.
    - More powerful models through careful methodology.

We will share slides and worked examples as R code:

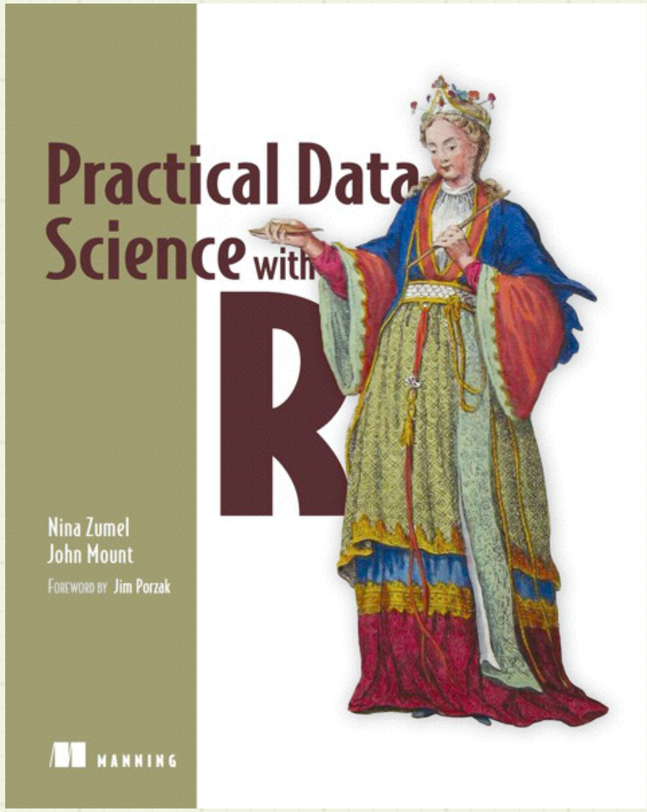
https://github.com/WinVector/NestedModelsTalk

Save the above URL and you have access to all links mentioned in this talk.

Who are we?

Nina Zumel
Win-Vector LLC
Dr. **Nina Zumel** is a principal consultant and founder at Win-Vector LLC a San Francisco data science consultancy and training company. Nina started her advanced education with an EE degree from UC Berkeley and holds a Ph.D. in Robotics from Carnegie Mellon University. Nina has worked as research scientist as SRI and developed revenue optimization platforms. She frequently writes and speaks on statistics and machine learning.

Nina is also the coauthor of the popular book of *Practical Data Science with R* (Manning Publications, 2014).



John Mount
Win Vector LLC
Dr. **John Mount** is a principal consultant and founder at Win-Vector LLC. John has worked as a computational scientist in biotechnology and a stock-trading algorithm designer and has managed a research team for Shopping.com (now an eBay company). John started his advanced education in mathematics at UC Berkeley and holds a Ph.D. in computer science from Carnegie Mellon.

John is also the coauthor of *Practical Data Science with R* (Manning Publications, 2014).

Please contact contact@win-vector.com for projects and collaborations.
http://win-vector.com/ Twitter: @WinVectorLLC .

Win-Vector LLC

Goals of this section

- Understand model quality evaluation.
    - Going to work with classification models.
- Give you a sophisticated tool box of model quality measures that are:
    - Statistically well founded.
    - Decisive.
    - Adapted for business and big data.
    - With ready to go R code and graphs.
- Look at models as data scientists, with emphasis on how we would explain them to non data scientists.

Section Outline

- Work on model scoring and significance of fit.
- Work on effect sizes.
- Work on model to model comparison.

Classification model

- A classification problem is a collection of pairs (x,y) where x (also called the independent variables) is what is known about an individual and y (called the dependent variable or outcome) is the class the individual belongs to.

- Example:

    - x = Recent shopping history of a web-site visitor.

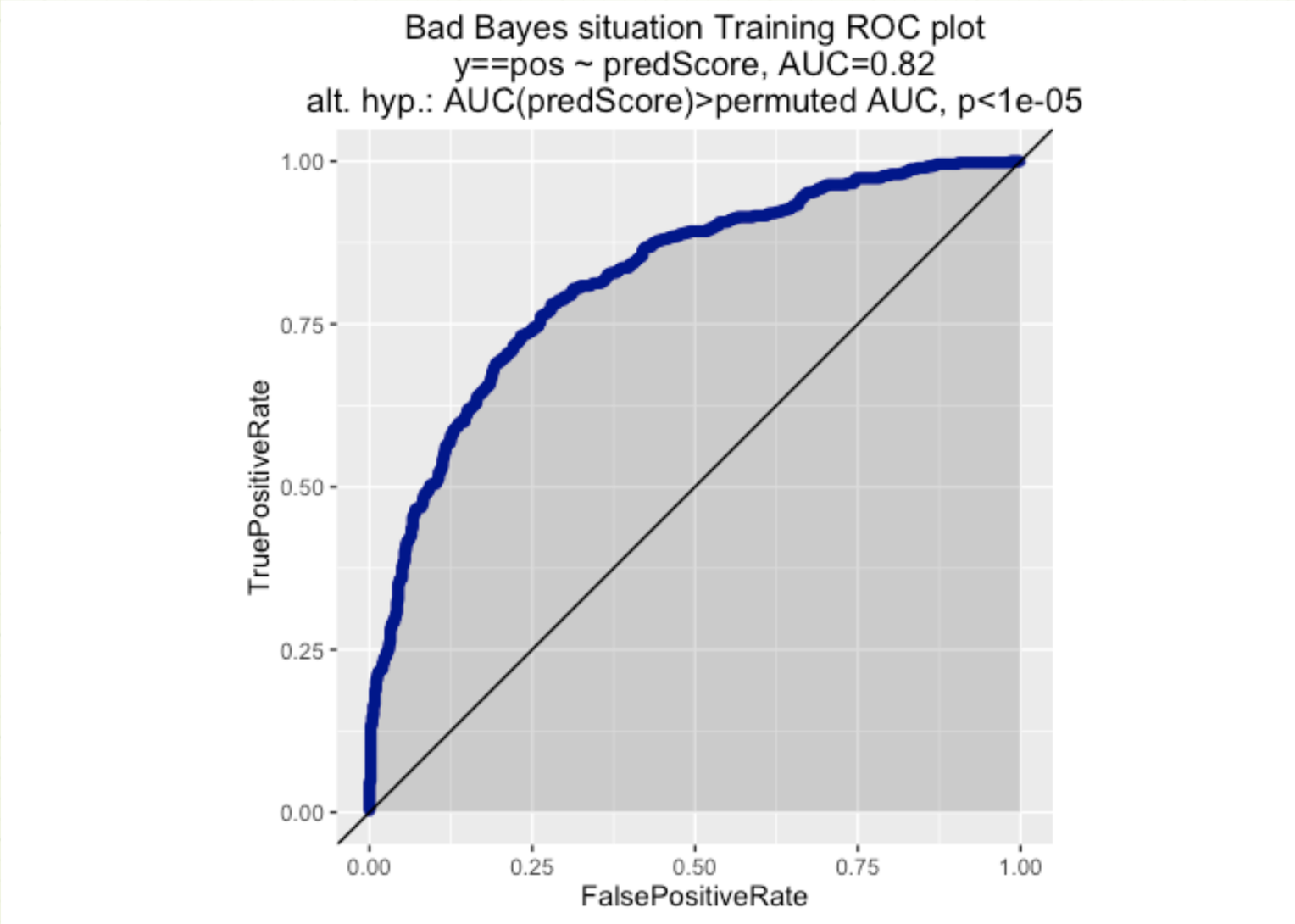
    - y = True if the click on our next advertisement, otherwise False.

- In past or training data we know pairs (x,y). In future or application data we know only x, and want the classification model to tell us y.

- We refine this to insist on a classification model is a function f() such that f(x) is the predicted score (or even better, predicted probability) that x is in the given class.

Do we have a good model?

- Model seems to work on training data.
  - Training data defined here as: data available to your data science team.



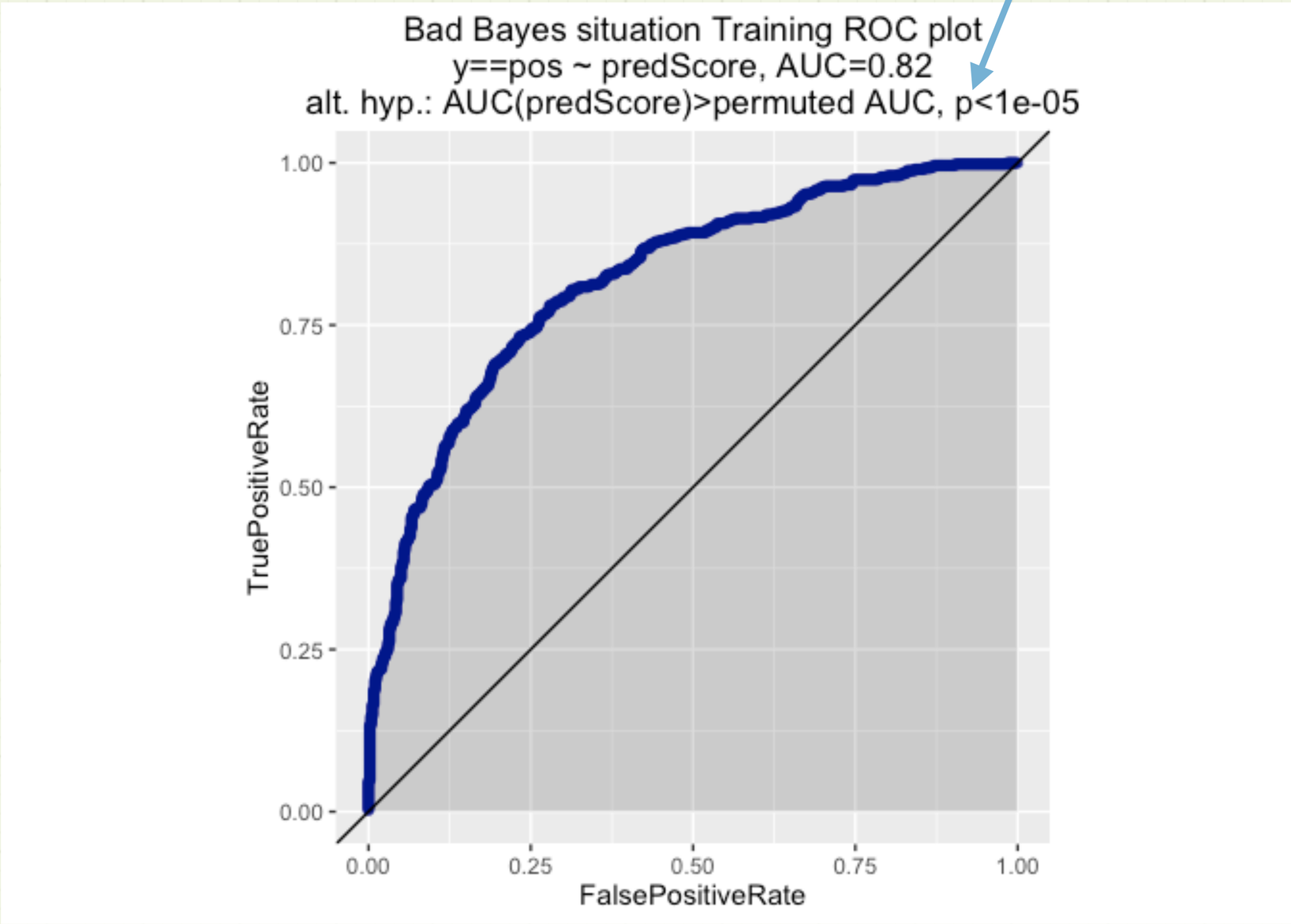
https://github.com/WinVector/ValidatingModelsInR/blob/master/Lab04PermTestVarSel/PermutationSelection.md

Win-Vector LLC

How to explain the curve?

"p" will be important going forward.

- It is called the "receiver operating characteristic" plot.
- The plot abstracts out the classifier score as an irrelevant detail.
- You can pick one of "true positive rate", or "false positive rate" and the curve tells you the other achievable by your classifier.
- AUC is "area under the curve" and is 1.0 for perfect scoring systems, and 0.5 or less for bad systems.



Significances / p-values

- Significance (or “p”) reads off the probability of a “null model” (a useless model you hope to reject or prove your model is not) can achieve a given performance statistic by chance on a given data set.
- It is a joint quantity depending both on the model and on the data, so unfortunately it is not a property of the model alone.
- Important to remember: it is a “one sided test.”
    - p near 0 *can be* good.
    - p near 1 *is* bad.
- *Lots* of confusion between coefficient p-values and model quality p-values.
- Not always reported by standard packages (even in R and Python’s scikit learn)

You should *always* expect at least a model quality p-value

- For example: linear regression summary reports one in R.

```
> summary(lm(y~x,data.frame(y=1:3,x=c(1,1,2))))

Call:
lm(formula = y ~ x, data = data.frame(y = 1:3, x = c(1, 1, 2)))

Residuals:
         1          2          3
-5.000e-01  5.000e-01  3.053e-16

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.282e-15  1.225e+00   0.000    1.000
x           1.500e+00  8.660e-01   1.732    0.333

Residual standard error: 0.7071 on 1 degrees of freedom
Multiple R-squared:   0.75,Adjusted R-squared:    0.5
F-statistic:     3 on 1 and 1 DF,  p-value: 0.3333
```

- p not near 0 is bad.
- It means your experiment was too small to measure an effect as weak as the one seen.

Win-Vector LLC

Unless you *insist*, you don't always get a model quality p

- Even R's glm neglects to supply one.

```
> summary(glm(y~x,data.frame(y=c(T,F,F),x=c(1,1,2)),family=binomial))
Call:
glm(formula = y ~ x, family = binomial, data = data.frame(y = c(T,
    F, F), x = c(1, 1, 2)))

Deviance Residuals:
       1         2         3
 1.17741  -1.17741  -0.00008

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    19.57   10754.01   0.002    0.999
x             -19.57   10754.01  -0.002    0.999

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3.8191  on 2  degrees of freedom
Residual deviance: 2.7726  on 1  degrees of freedom
AIC: 6.7726

Number of Fisher Scoring iterations: 18
```

Remember this word.

No p.

????????

Expect even worse reporting with sophisticated machine learning packages

Random reddit thread as “authority”:

https://www.reddit.com/r/pystats/comments/3c6idg/sklearn_stderrorspvaluesttest/

Sklearn stdErrors/p-values/t-test (self.pystats)
submitted 1 year ago by codingpynoob
Im[sic] trying to use sklearn for some Linear Regression and I cannot for the life of me find what methods I can call to get this information. am i just going to have to calculate it myself?
Compared to statsmodel, sklearn seems really painfully in being able to evaluate the linear regression models your[sic] building

[–]**BeatLeJuce** 3 points 1 year ago*
There are no such methods in sklearn, as the package is more meant for applied ML, and treats algorithms as black box models that you can swap in/out as needed. This is in contrast to typical "statistical" treatment of Regression where you calculate all sorts of statistics and try to interpret your model as best you can. If you need significance measures for your coefficients you'll have to either calculate them yourself or use statsmodel.

Unified view: compute significance outside the modeling procedure.

- We are starting such a package with *sigr*
  - https://github.com/WinVector/sigr

```
> sigr::formatFTest(lm(y~x,data.frame(y=1:3,x=c(1,1,2))),pLargeCutoff=1)$formatStr
[1] "F Test summary: (R2=0.75, F(1,1)=3, p=0.33)."
```

```
> sigr::formatChiSqTest(glm(y~x,data.frame(y=c(T,F,F),x=c(1,1,2)),family=binomial),pLargeCutoff=1)$formatStr
[1] "Chi-Square Test summary: pseudo-R2=0.27 (X2(1,N=3)=1, p=0.31)."
```

What you make easy tends to actually get done.

Win-Vector LLC

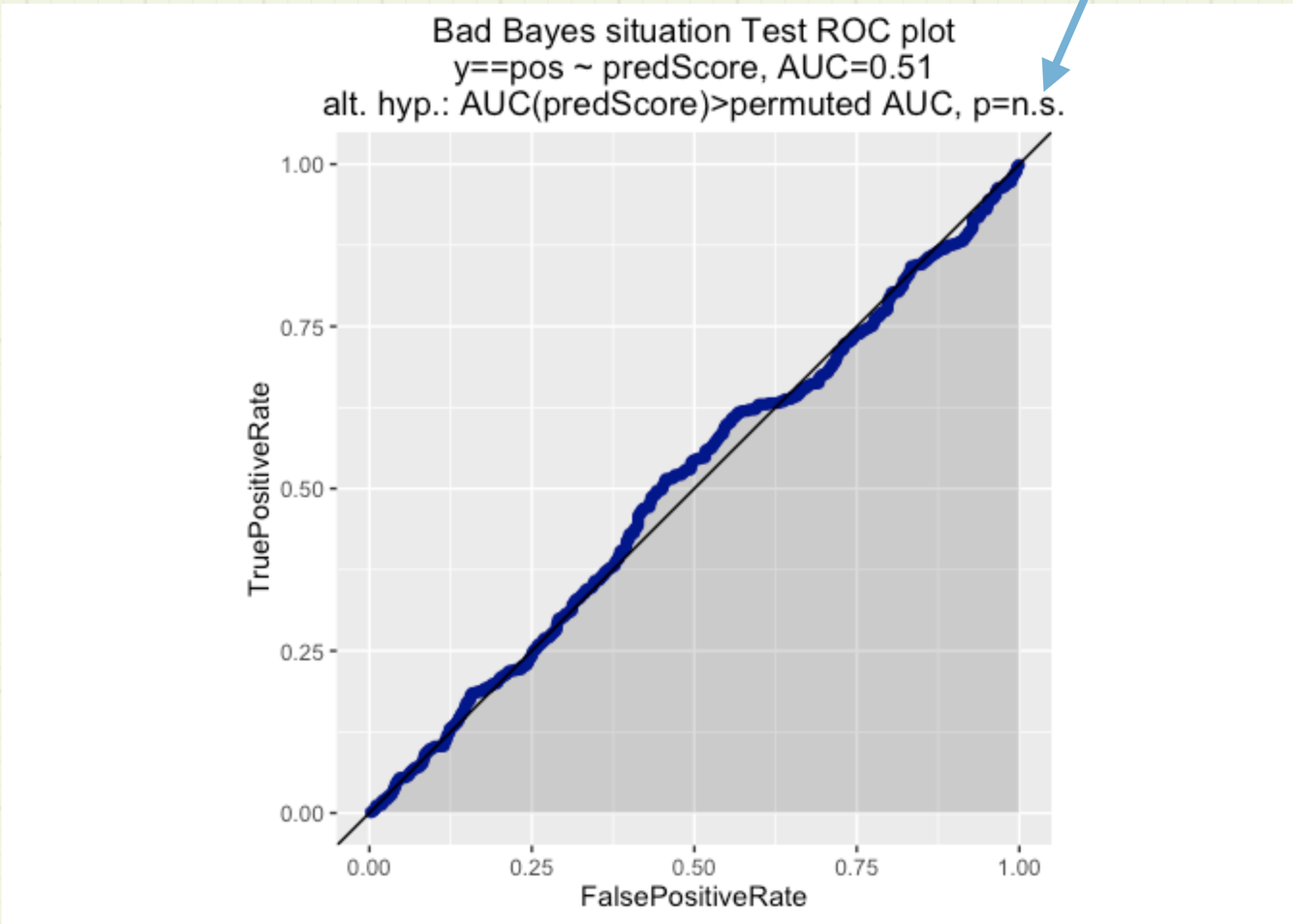
If you take away only one thing …

- If you take away only one thing about hypothesis testing from this section (or unlearn only one fallacy) please let it be this:
  - **You can not "pass a frequentist significance test."** (especially on training data)
  - This is what we mean't by calling the test "one-sided."
  - Please see: http://www.win-vector.com/blog/2016/10/the-unfortunate-one-sided-logic-of-empirical-hypothesis-testing/ for some ranting on this topic.
- What you want is a good estimate of the model's future performance on application data.
  - This is *essentially* a Bayesian question
  - http://www.win-vector.com/blog/2013/05/bayesian-and-frequentist-approaches-ask-the-right-question/

Q?: is the model *really* predicting *anything* at all?

"n.s" is shorthand for "not significant" or p near 1.

- Had a nice small p on training.
- Model fails miserably in production.
- Or (better because it means you find the problem earlier) fails on held-out test data.



https://github.com/WinVector/ValidatingModelsInR/blob/master/Lab04PermTestVarSel/PermutationSelection.md

Win-Vector LLC

Why does this happen?

- This is called “overfitting” or “excess generalization error.”

- Often the modeling procedure is so powerful that it can identify and use *apparent* predictive relations, when in fact there are in fact no such *actual* relations.

- With “wide data” (very many columns being considered as variables) this can happen to most state of the art classifiers (random forest, boosting, logistic regression, and so on). *Even* those that include cross-validation procedures as part of their design.

  - https://github.com/WinVector/PreparingDataWorkshop/tree/master/BadModels

How do we detect overfit?

1. First method: re-evaluated model on data not made available during model construction.

   - A hold-out or lockbox data set.

2. Second method: cross validation procedures.

   - Simulate access to new data, by training on less data.

3. Third method: permute the outcomes or “y”s and see how well the modeling procedure does.

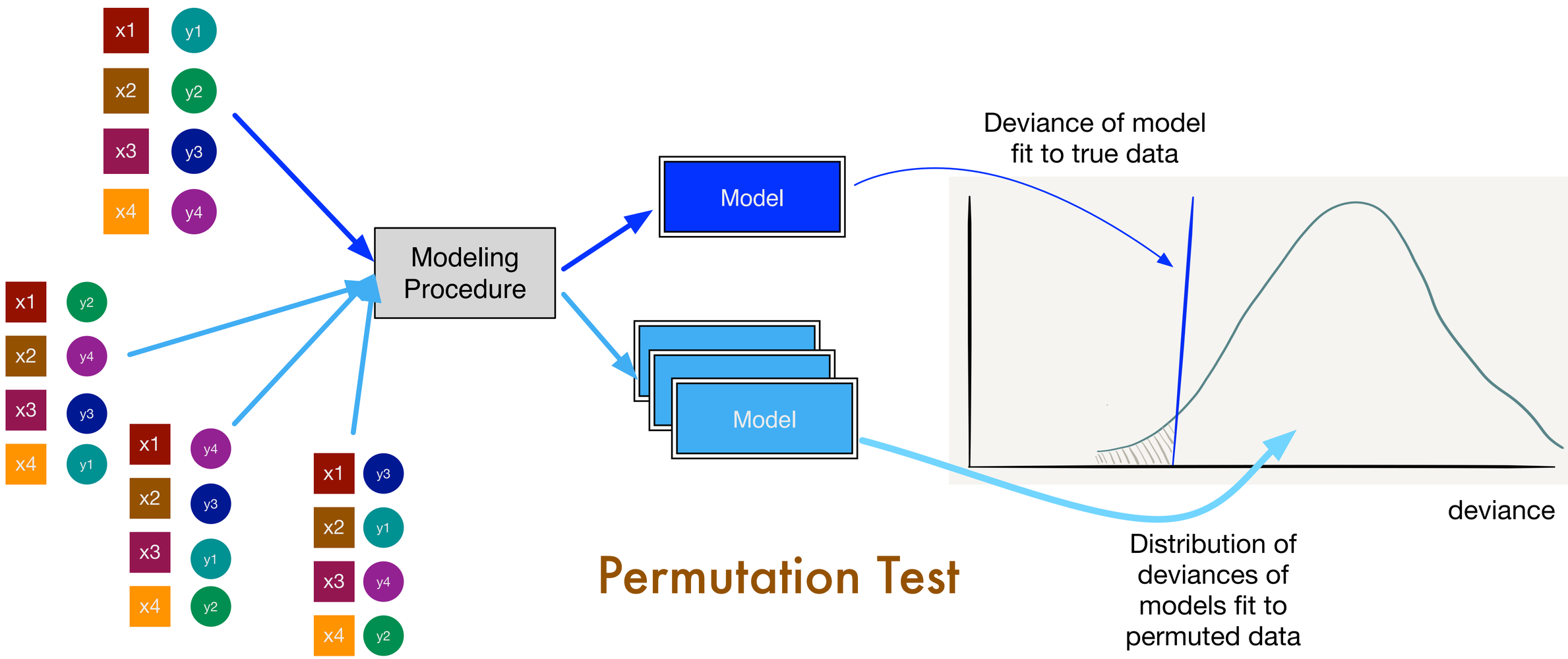
   - Catch the technique in a lie.

Example: deviance score

- We will demonstrate the permutation method on a different score called “deviance.”
- Deviance is “expected log-embarrassment” or in terms of entropy: how much residual surprise is in the data, given the model’s predictions.
    - -2 sum_i log(predicted_probiblity[y_i])
        - If y_i is “True” deviance contribution is -2 log (f(x_i))
            - Wants f(x_i) near 1 when y_i is “True.”
        - If y_i is “False” deviance contribution is -2 log (1 - f(x_i))
            - Wants f(x_i) near 0 when y_i is “False.”
        - Prefect score is zero (where f(x_i) is 1 if y_i is “True” and 0 otherwise).
        - Larger scores are worse
- Is the score the glm/logistic-regression model should have given us a p-value for.

Win-Vector LLC

Permutation Thought Experiment: Is the input *really* related to the output?

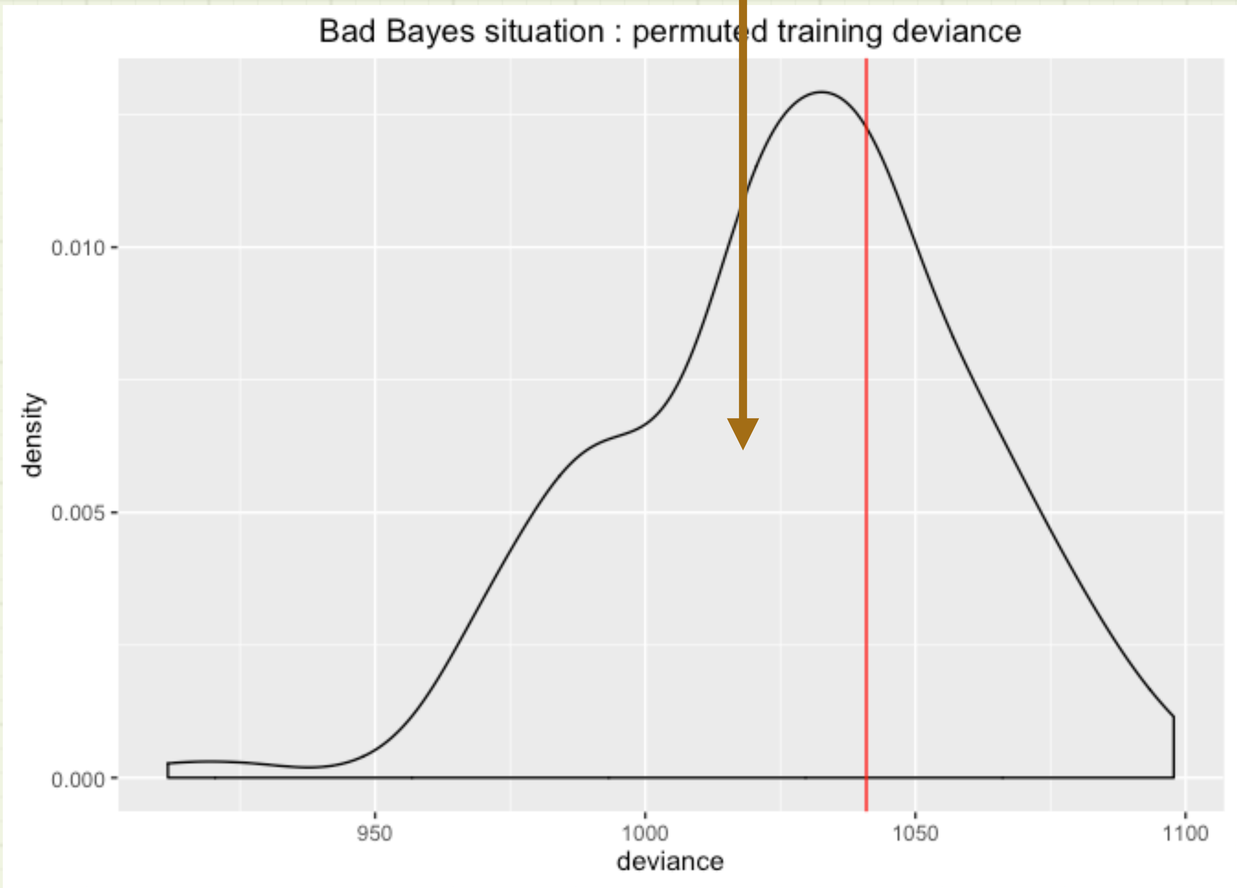


Our example again

Probability that model on a noise set would do better than our model: model significance

- In our permutation experiment we saw a deviance as good as or better (less) than our claimed fit 66% of the time!

- The claimed fit does not stand out, it isn't evidence of true signal.



Win-Vector LLC

What are these scores again?

- AUC: “area under the curve”
  - The probability that a classifier will rank a randomly chosen true instance higher than a randomly chosen false one, with 1/2 point for ties.
  - Detects if any re-shaped or re-scaled version of the model score would be useful.
  - Very good metric early in a project. Used a lot in Kaggle data science competitions.
- deviance: “expected log-embarrassment”, “residual surprise”
  - Penalizes claiming events that happened were rare.
  - Good double check of model calibration before going into production.
- Both attempt to measure quality of the score independent of the intended application.

On scores

- There are a lot more scores than “accuracy.” Often a business partner uses the word “accuracy” as a stand-in or synonym for any number of quality measures.
- Let your data science team start with AUC and deviance.
- Have them move on to business relevant scores (such as expected return on investment) as you get closer to project deployment.
- Successful data science projects may start as machine learning projects (where technical metrics are most useful), but they end as business services (where business metrics are most useful).

Review: we can easily directly test if we have a good model in R (or any other scriptable analysis toolkit).

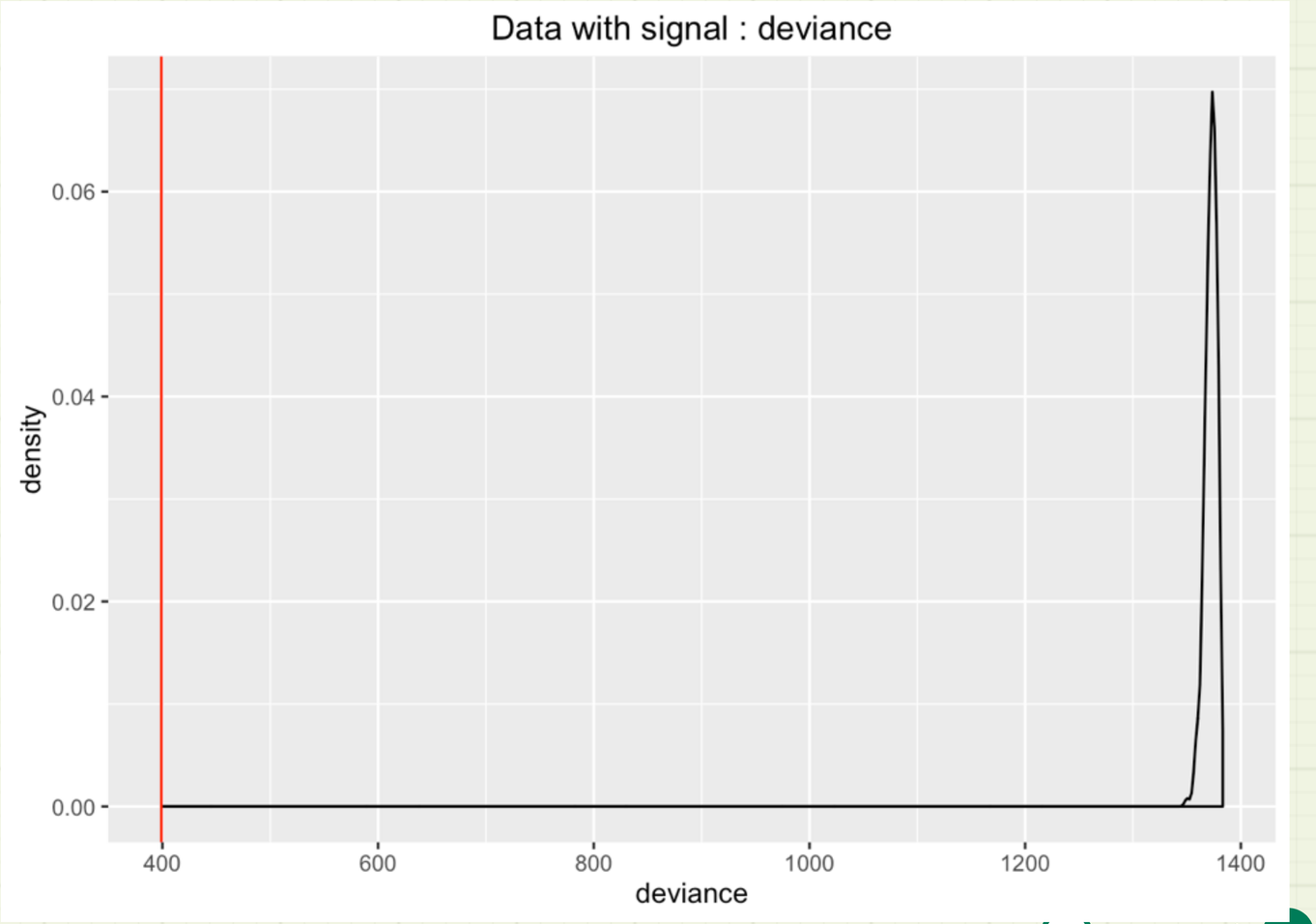
Ex G: what a good model and measurement looks like

```
##   deviance                          label
## 1 399.0531 Data with signal Training
## 2 414.5022     Data with signal Test
```

Again: How do we know if “400” is a good deviance or not?

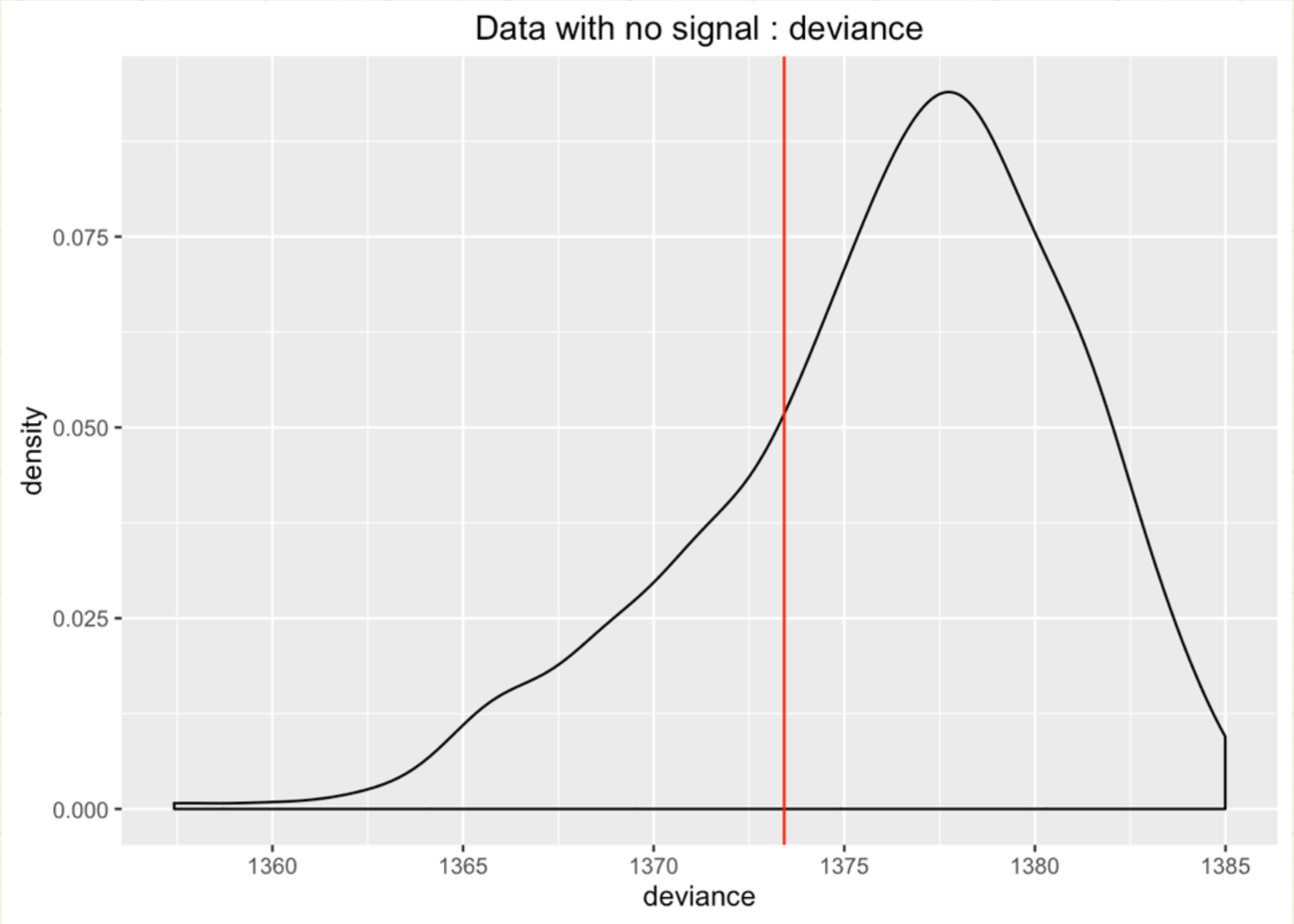
Ex G: permutation test establishes scale

```
##   deviance                      label
## 1 399.0531 Data with signal Training
## 2 414.5022     Data with signal Test
```



Ex B: what a bad measurement looks like

```
##   deviance                          label
## 1 1373.424 Data with no signal Training
## 2 1395.741     Data with no signal Test
```



Note: "overfit" is not the only potential issue.

Win-Vector LLC

Two additional issues

- Data science projects tend to have large sample size.
- We often don’t have the advantage of deploying the first ever model, but instead proposing a challenger model to improve on a current champion.

Data science projects tend to have a large sample size

- This is good.
  - A lot of the hard parts of statistics are trying to get delicate calculations right for small samples.
- This can be too good.
  - The classic “z and p” statistical formulations are not targeted at the big data world.

“z and p” statistics

- In the classic z and p formulation you estimate significance in three steps.

  1. Convert your question into a sample statistic on a population (ex: average per-row deviance of a classifier on a new test population).

  2. Summarize this statistic and compute z = “how many population standard errors from the non-informed mean value your measure is.”

  3. Use the appropriate distribution table (normal, Student’s-t, chi-square, …) to convert the z into a significance p.

     - For deviance we used the chi-square test.

     - For AUC we use a Studentized bootstrap test.

Behavior of z and p on large populations

- As sample size goes to infinity …
  - If there is no effect (our model is the same as a null-model) then:
    - z is roughly normally distributed with standard deviation 1.
    - p is uniformly distributed in the interval [0,1], *independent of experiment size*!
    - Different experimenters do not agree on z and p!
  - If there is an effect (our model is better than the null-model, by any fixed amount no matter how small).
    - z goes to infinity.
    - p goes to zero.
    - Without additional facts (such as experiment size) all non-negligible effects have the same qualitative behavior!

An alternative

- Cohen's "effect size index" d.
  - d is the effect size divided by the standard deviation of individuals (not the standard deviation of the population mean as with z).
    - We can (and should) compute error-bars and significances on the location of our estimate of d.
  - d = z / sqrt(npop).
    - d is dimensionless, but has a built in interpretation.

From *Statistical Power Analysis for the Behavioral Sciences*, 2nd edition, Jacob Cohen, Lawrence Erlbaum Associates, 1988.

small: **d** = .20,
medium: **d** = .50,
large: **d** = .80.

Cohen's d example

- Population: heights of 19 year old US males and females in centimeters.

sex	n	mean	seOfMean
Male	179	177.8	0.81
Female	118	163.4	0.72

- Population means separated by over 20 standard deviations.
- Cohen's d: 1.49. Consistent with the difference between the height of a man and a woman being about 1.46 times the height difference between two individuals with the same sex.
- Relation: E[(male-female)^2]/E[(male1-male2)^2] - 1 ~ (1/2)*(Cohen's d)^2
- Think of Cohen's d as a measure of excess difference.

https://github.com/WinVector/ValidatingModelsInR/blob/master/Cohensd/heights.md

Win-Vector LLC

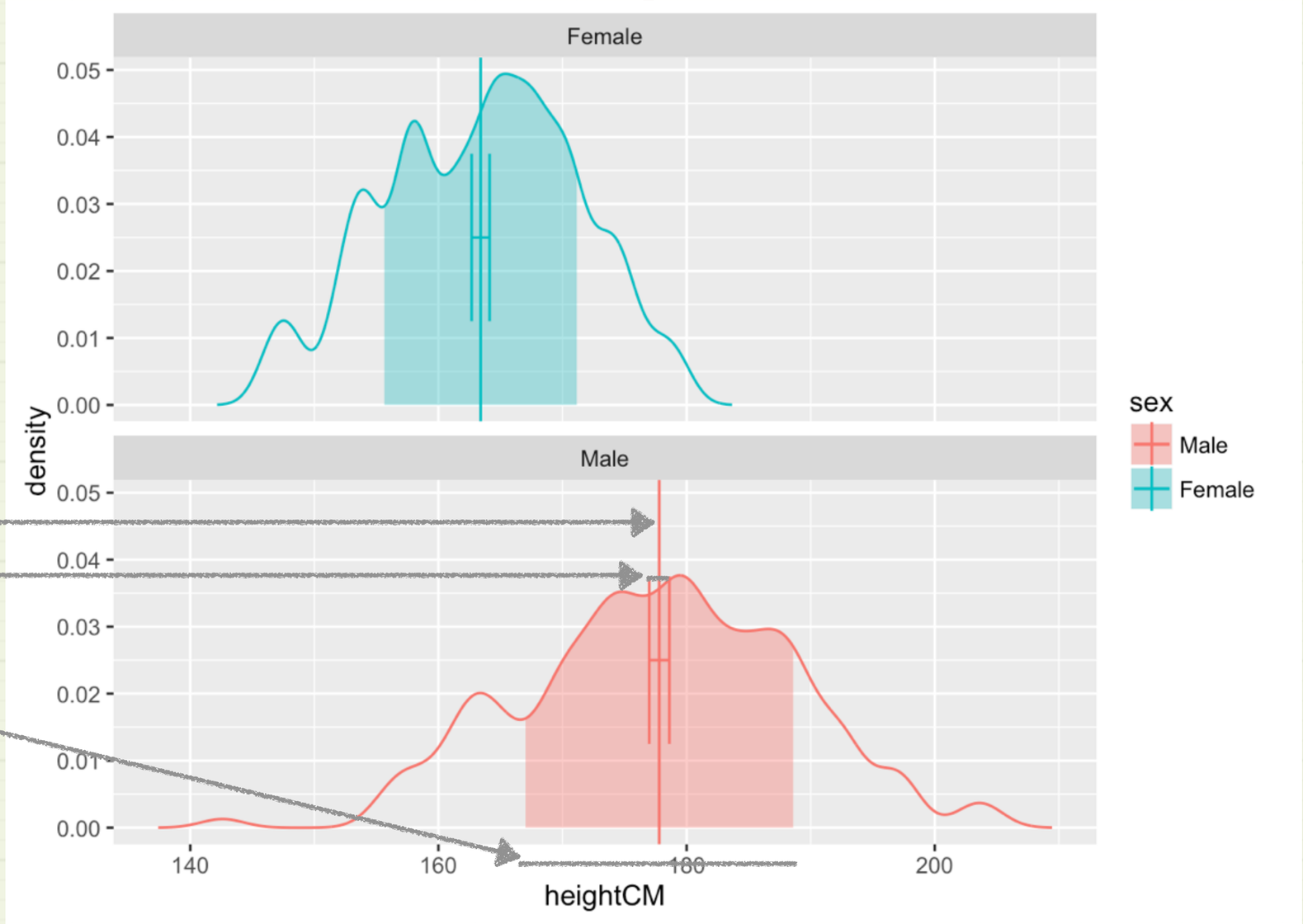
The quantities in question

- Should *always* report effect size.
- In fact you should report it at least twice.
  - Once in domain units (such as centimeters).
  - Once in standard deviation units (such as Cohen's d).

mean

standard error of mean

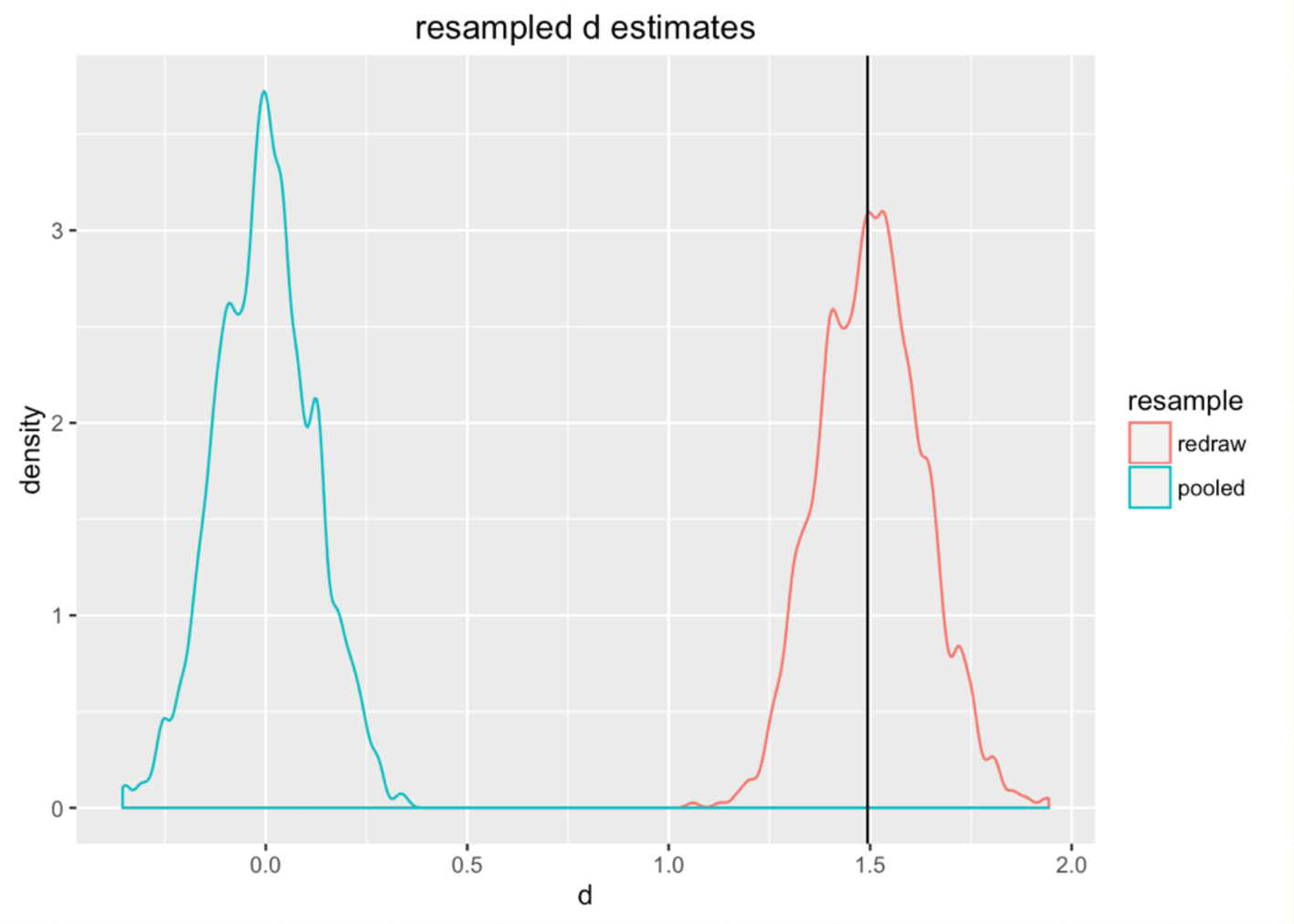
standard deviation of individuals



Note:
For a,b i.i.d. E[(a-b)^2] = 2 E[(a-E[a])^2], so we expect individual to individual differences to be about sqrt(2) larger than the standard deviation of the individuals.

Win-Vector LLC

Of course we should go beyond a point estimate



- With some thought it should be possible to know what an estimated d might mean.
  - It indicates if we are studying a large or small effect.
- This is separate from the question of have we accurately, reliably or precisely estimated d.
  - For that we need distributional details such a error bars, significances, or even distributional plots.

Win-Vector LLC

Review: how Cohen's d works

- As sample size goes to infinity …
  - If there is no effect (our model is the same as a null-model) then:
    - d converges to zero (with shrinking error bars).
  - If there is an effect.
    - d converges to a constant proportional to the effect size (with shrinking error bars).
  - Different good experiments should agree on d!

Some unfortunate details

- Deciding how you pool variance, deal with sample size, and such explodes Cohen's d into many related measures (such as Hedges' g).
- Adding in details of inference (moment matching, maximum likelihood, and how significances are estimated) further metastasizes the issue.
- This is why you have dozens of measures of correspondence (see http://www.win-vector.com/blog/2016/07/a-budget-of-classifier-evaluation-measures/) and hundreds of relevant significance tests (for example: *100 Statistical Tests,* Gopal K. Kanji, Sage Publications, London, 1999).

Our advice

- Prefer a few empirical tests.
- Prefer tested implementations:
  - https://CRAN.R-project.org/package=pwr
  - https://CRAN.R-project.org/package=boot
  - https://github.com/WinVector/sigr
  - https://CRAN.R-project.org/package=cvAUC
  - https://github.com/WinVector/WVPlots
  - https://CRAN.R-project.org/package=vtreat
- More specific tests have computational (run faster) and inferential (are more correct) advantages, but take time to master and explain to partners.

Aren’t we ignoring details?

- Bootstrap Cons:
  - A better person *would* know the closed form for the significances of all of these statistics.
  - Empirical re-sampling has issues with rare events and non-smooth statistics (such as number of unique values in a sample).
  - A “statistic” is defined as a deterministic function of a sample, you can’t formally treat non-deterministic *estimates* of things like Cohen’s d as if they were in fact *statistics*.
- Pros:
  - With empirical methods you can use the summary statistics you want, not just those you have a sufficient reference for.
  - Most data science quality scores have a nice additive interpretation.
  - Most of the ignored corrections are monotone- so you would sort most things in the same order and make the same decisions (but with different thresholds).

Example: AUC

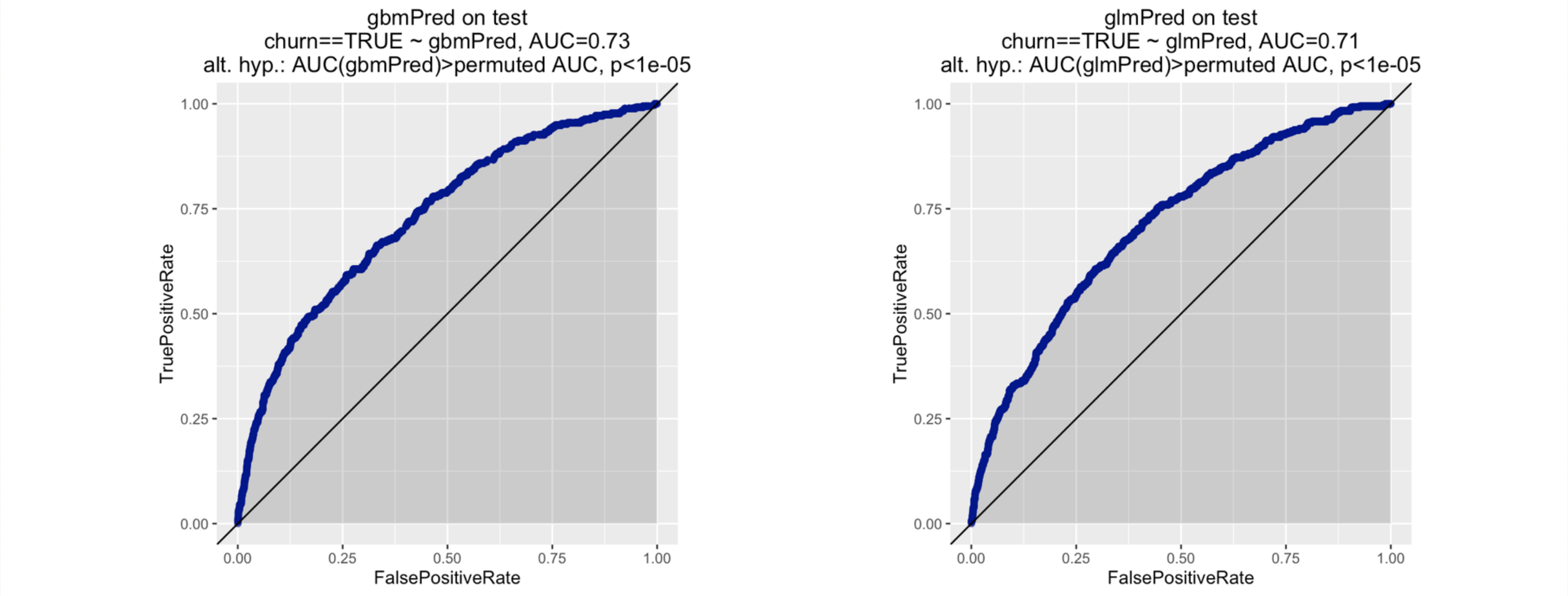
- At first glance depends on whole scored set in a non-additive way.
- With some work discover it is additive over pairs of observations (counting number of correct orderings induced by the model score, with appropriate tie-breaking).
- Therefore can use the Wilcoxon–Mann–Whitney U-test to compute significance.
- Aaannnd that doesn't matter.
    - For large samples, U is approximately normally distributed.
    - From R's stats::wilcox.test:
        - "`This function can use large amounts of memory and stack (and even crash R if the stack limit is exceeded) if exact = TRUE and one sample is large (several thousands or more).`"

Win-Vector LLC

Actually, not all questions are easy

- Often data scientists are asked to measure very small differences.
- This uses up a lot of the big data advantage.
- To overcome this *must* rephrase questions away from “which model is best” (which can become infinitely expensive to answer) towards “what is the expected loss?”

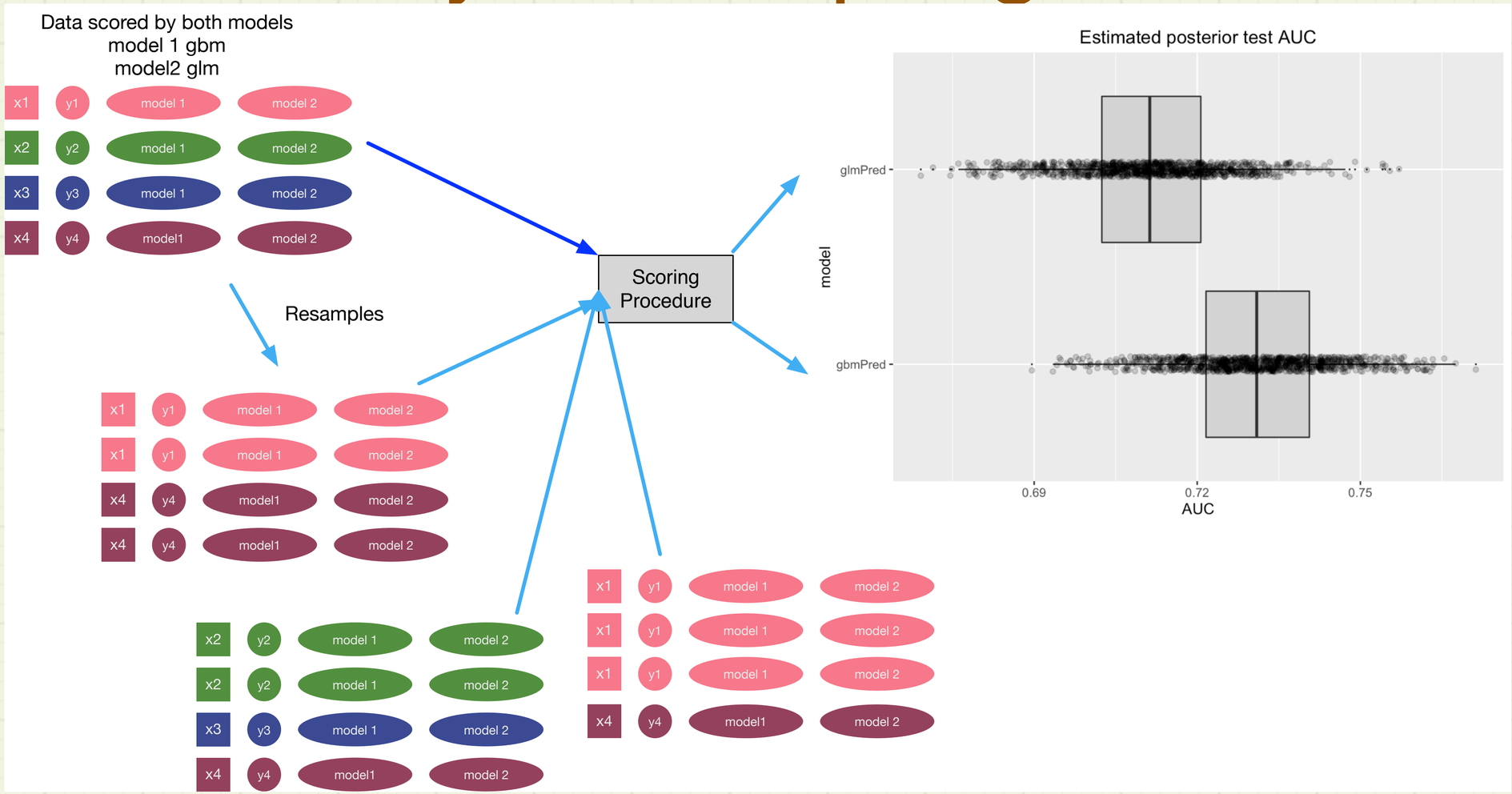
Example: two KDD2009 contest models on test data



Is such a change in AUC important/significant/repeatable?

Win-Vector LLC

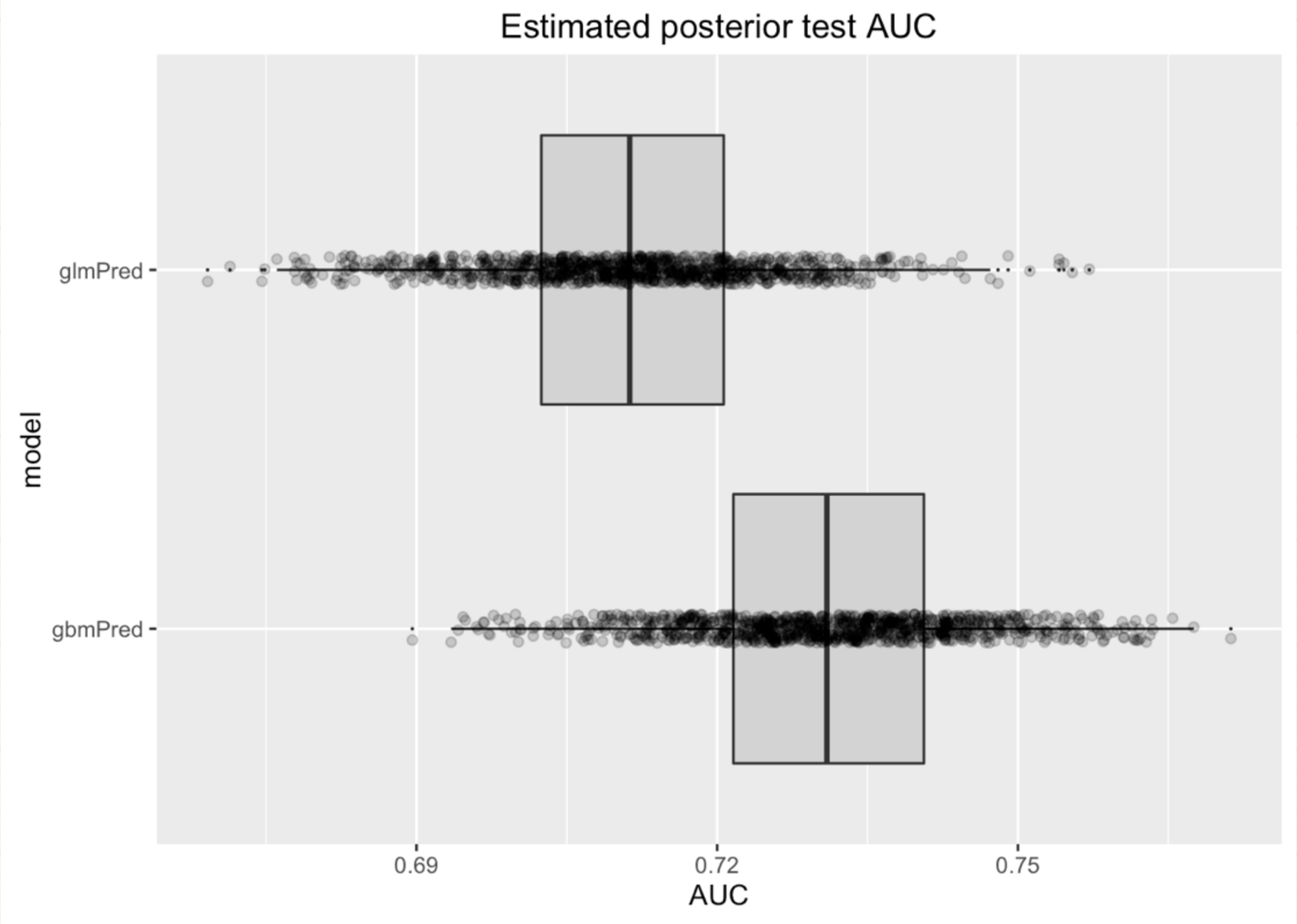
Simulate Both Models on “future data” by re-sampling test rows



Empirical re-sampled (with replacement) scoring

Win-Vector LLC

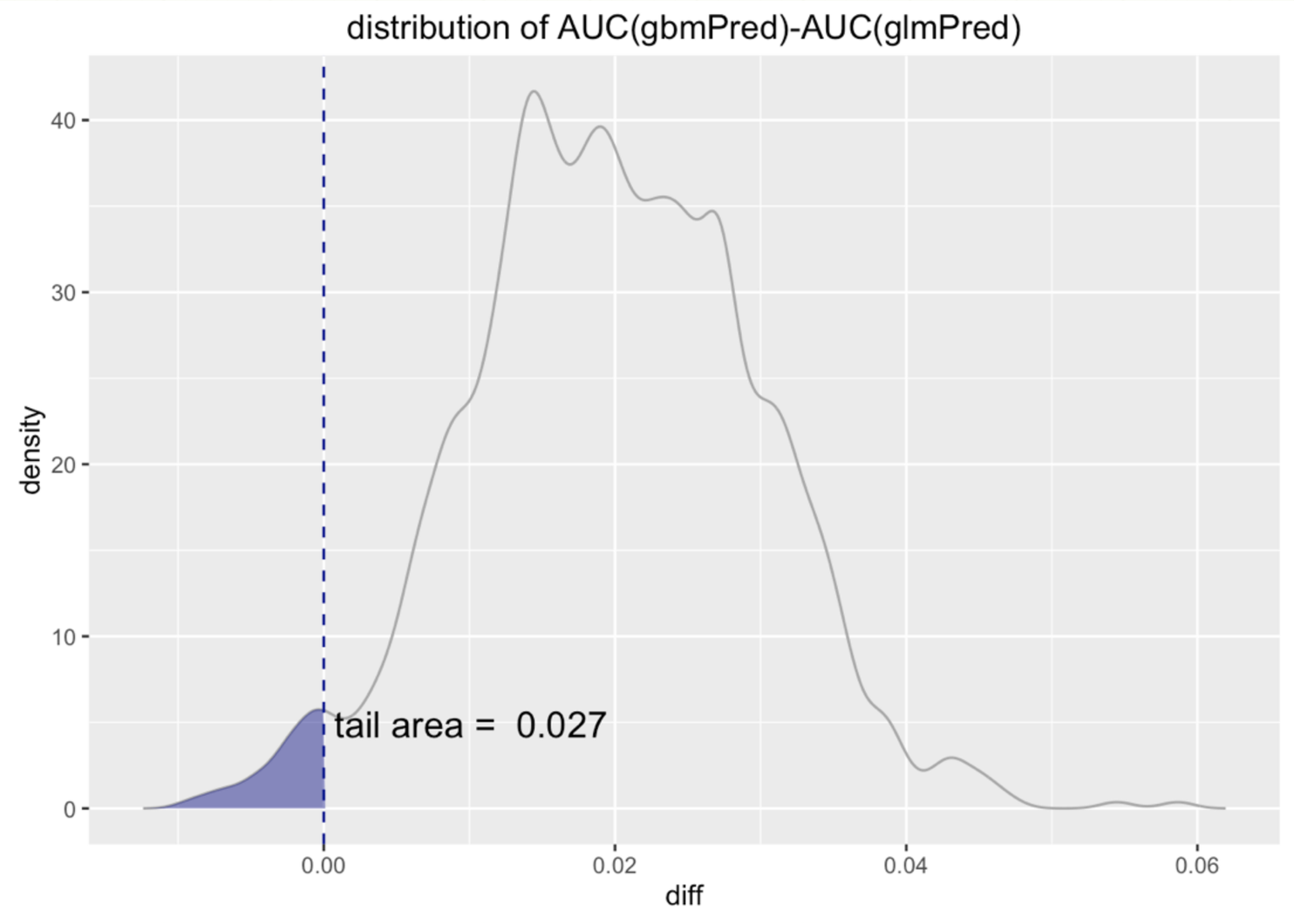
“e” based on a joint bootstrap sample



Really only a posterior under the fairly unconvincing uniform prior.

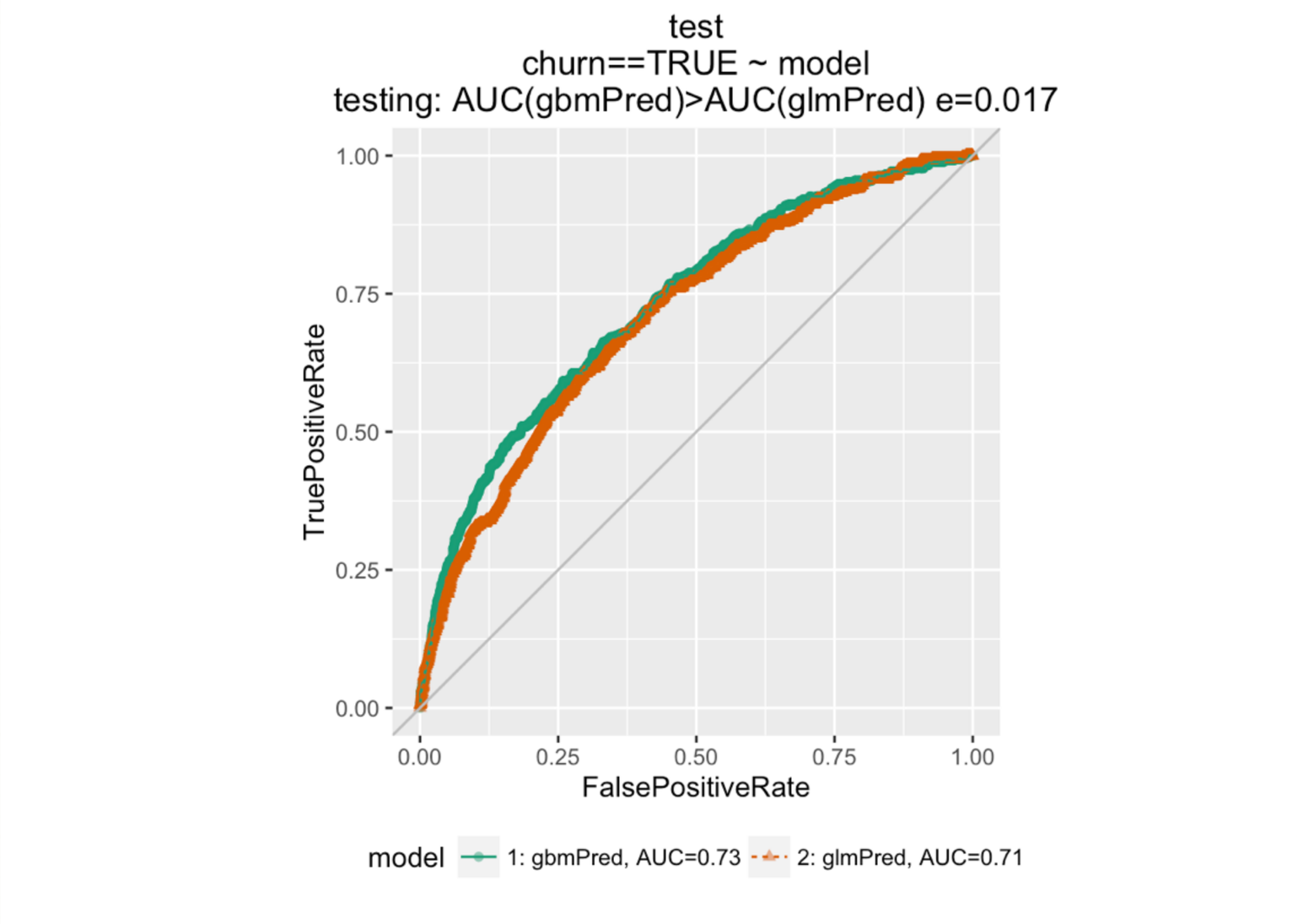
Win-Vector LLC

This sample gives an empirical probability of glm outperforming gbm



Win-Vector LLC

Can build a comparative ROC plot



- e (error rate) is an estimate of how often the glmPred model outperforms the glmPred model on re-samples of the scored test set.
- This reported e is "Studentized", so not identical to exact observed error rate.

Win-Vector LLC

Great technique, wrong question if you are not the first model

- In industry nobody wants to waste their test infrastructure time building confidence intervals around nearly indistinguishable models.

- The amount of data needed to reliably measure an effect of size epsilon is usually about 1/epsilon^2, which goes to infinity very fast as epsilon goes to zero.

- The methodology is right, but we want to concentrate on experiments that show large possible wins or opportunities.

Better: compute "opportunity at risk"

- Let A and B be two classifier inducing scores A_i and B_i (assume larger scores are better).
- Define OpportunityAtRisk(A,B) as:
  - E_i[max(0, score(B_i) - score(A_i))].
  - It is the error rate weighted by the "hinged" magnitude of the difference in scores.
- OpportunityAtRisk(A,B) goes to zero as B approaches A.

Opportunity at risk for our example gbm/glm example

- We estimated:
  - AUC(gbmPred) = 0.732
  - AUC(glmPred) = 0.711.
  - OpportunityAtRisk(gbmPred,glmPred) = 7.4e-5
  - OpportunityAtRisk(glmPred,gbmPred) = 0.02
- If we choose gbmPred as our classifier the opportunity at risk for our choice is 7.4e-5.
- Diagnostic: large chosen opportunity at risk (or low resampling variances) would indicate a larger experiment is needed.

Which model would I pick?

- gbm
  - If I were only worried about overall classification or sorting performance, I would pick gbm.
  - If I use my experience as data scientist as a prior: I would go with gbm, as it *is* often better than glm.
- However, glm has a number of advantages beyond questions of predictive performance:
  - It is computationally more efficient (runs faster)
  - Logistic regression is a lower variance fitting procedure than gradient boosting, which means, among other things, it is less sensitive to slight variations in the training set distribution.
  - Logistic regression models are more interpretable
- Or may I would stack the models using a super learner and get a model better than either of these.

Conclusions / Take aways

- Model evaluation needs to be integrated into the entire data science process, including project proposal and result delivery.
- R gives a flexible, highly visual framework that implements advanced machine learning, classical statistical tests, important empirical tests, and any necessary ad-hoc capabilities.
- Can add statistical checks to arbitrary machine learning procedures.

Some More Reading

- “How do you know if your model is going to work?”
  - **http://www.win-vector.com/blog/2015/09/isyourmodelgoingtowork/**
- “How Do You Know if Your Data Has Signal?”
  - **http://www.win-vector.com/blog/2015/08/how-do-you-know-if-your-data-has-signal/**
- “Statistics to English Translation”
  - **Part 1: Accuracy Measures**
  - **Part 2a: ’Significant’ Doesn’t Always Mean ’Important’**
  - **Part 2b: Calculating Significance**
- More testing in R:
  - **Finding the K in K-means by Parametric Bootstrap**
- Win-Vector LLC ODSC 2015 “Preparing Data workshop”
  - **https://github.com/WinVector/PreparingDataWorkshop/**
- "Validating Models in R" Strata 2016 RDay
  - **https://github.com/WinVector/ValidatingModelsInR**

Win-Vector LLC

Thank you

- Please stay in touch:
  - **@WinVectorLLC**
  - **http://www.win-vector.com/**
  - Please invite us to present to your data science or management teams!
- Next up:
  - Building more powerful models using nested models and simulated out of sample data.

Win-Vector LLC