Simulation and ROC

Yun (Renee) Zhang, PhD

Email: zhangy@jcvi.org

Monthly Biostatistics Discussion

02-20-2019

Outline

1. Random Number Simulation
2. ROC Curve and AUC
3. Examples
    I. Logistic Regression
    II. Hypothesis Testing

Random Number Simulation

What is simulation?

- Simulation is an analogy of "experiment" in a dry lab
- With simulations, the statistician **knows and controls the truth**
- Simulation is a procedure that relies on random numbers

- Video by Roger Peng on Simulation: https://www.youtube.com/watch?v=tvv4IA8PEzw

Probability Distributions in R

- Functions for the Normal probability distribution in R
  - `rnorm:` generate random Normal variates with a given mean and standard deviation
  - `dnorm:` evaluate the Normal probability density with a given mean/SD at a point (or vector of points)
  - `pnorm:` evaluate the cumulative distribution function for Normal distribution
  - `qnorm:` evaluate the quantile function for Normal distribution
- Similarly, generating random numbers from other distributions
  - `rpois:` generate random Poisson variates with a given parameter
  - `rgamma:` generate random Gamma variates with given parameters
  - `rbeta:` generate random Beta variates with given parameters
  - `runif:` generate random Uniform variates with given parameters

Probability Distributions in Python

```
from scipy.stats import norm
data_normal = norm.rvs(size=, loc=, scale=)
from scipy.stats import poisson
data_poisson = poisson.rvs(mu=, size=)
from scipy.stats import gamma
data_gamma = gamma.rvs(a=, size=)
from scipy.stats import expon
data_expon = expon.rvs(scale=, loc=, size=)
from scipy.stats import uniform
data_uniform = uniform.rvs(size=, loc=, scale=)
from scipy.stats import binom
data_binom = binom.rvs(n=, p=, size=)
from scipy.stats import bernoulli
data_bern = bernoulli.rvs(size=, p=)
```

rnorm() and set.seed()

- rnorm(n, mean = 0, sd = 1)

```
> x <- rnorm(10)
> x
 [1] -0.8204684  0.4874291  0.7383247  0.5757814 -0.3053884  1.5117812
 [7]  0.3898432 -0.6212406 -2.2146999  1.1249309
> x <- rnorm(10,20,2)
> x
 [1] 19.91013 19.96762 21.88767 21.64244 21.18780 21.83795 21.56427
 [8] 20.14913 16.02130 21.23965
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  16.02   20.01   21.21   20.54   21.62   21.89
> sd(x)
[1] 1.762901
```

- Set the random seed with `set.seed()` to ensure reproducibility

```
> set.seed(1)
> rnorm(5)
[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
> rnorm(5)
[1] -0.8204684  0.4874291  0.7383247  0.5757814 -0.3053884
> set.seed(1)
> rnorm(5)
[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

Random Seed

- **Always set a random seed when conducting simulation!**
- In R:

```
set.seed()
```

- In Python:

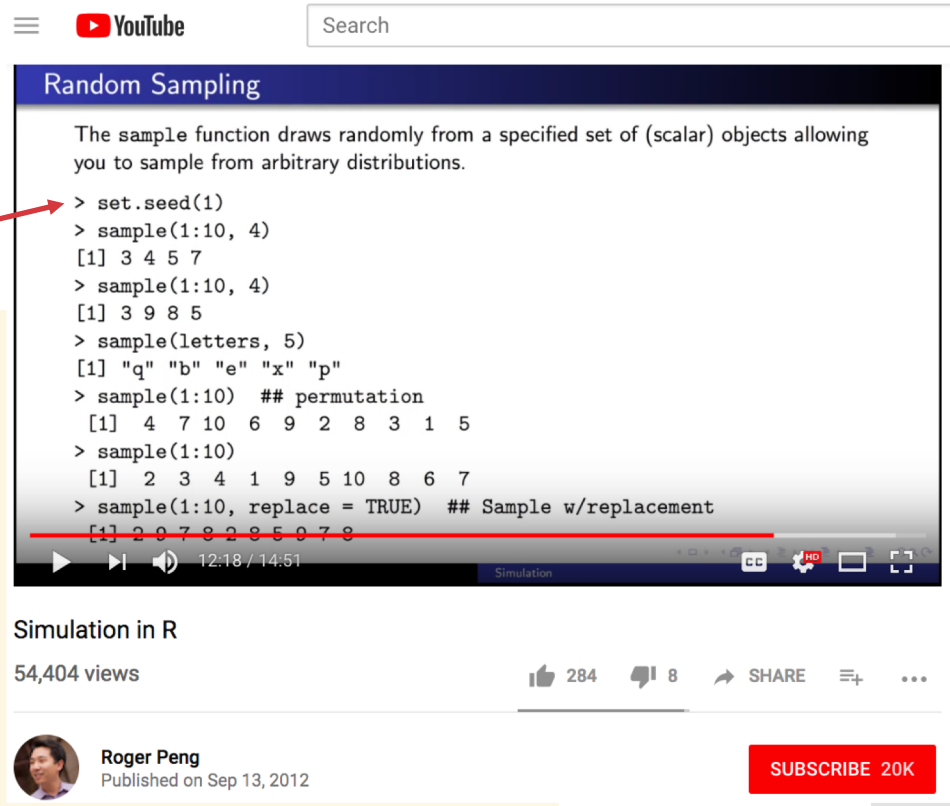
```
import random
random.seed()
```

Random Sampling

- The `sample()` function draws randomly from a specified set of (scalar) objects allowing you to sample from arbitrary distributions

same seed

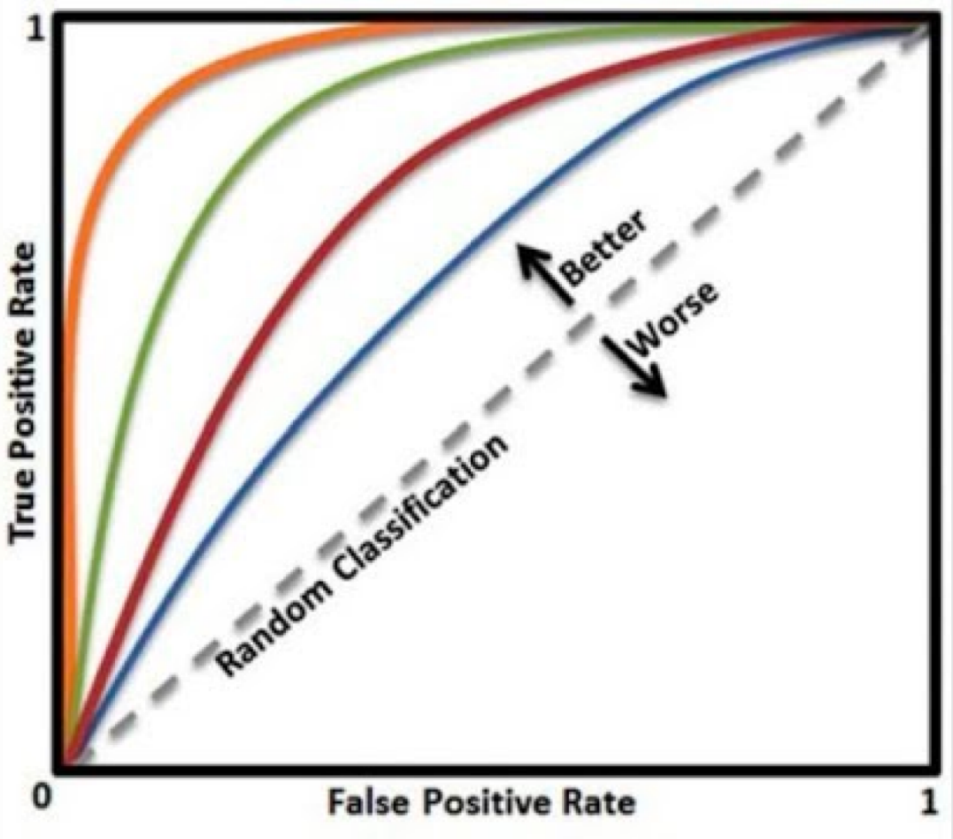
```
> set.seed(1)
> sample(1:10, 4)
[1] 3 4 5 7
> sample(1:10, 4)
[1] 3 9 8 5
> sample(letters, 5)
[1] "q" "b" "e" "x" "p"
> sample(1:10) #permutation
 [1]  4  7 10  6  9  2  8  3  1  5
> sample(1:10)
 [1]  2  3  4  1  9  5 10  8  6  7
> sample(1:10, replace=TRUE) #sample w. replacement
 [1] 2 9 7 8 2 8 5 9 7 8
```



Receiver Operating Characteristic (ROC) Curve

ROC curve

- ROC curve is a comprehensive visualization tool that evaluates the performance of one or more binary classifiers
- TPR vs. FPR
- Sensitivity vs. 1 – specificity
- Top-left corner = the best
- Diagonal = random guesses
- E.g. the orange classifier is *uniformly better* than all other classifiers; the orange curve *dominates* other curves



Confusion Matrix

- True positive rate (TPR) = true positives / all positives = Sensitivity
- False positive rate (FPR) = false positives / all negatives = 1 − Specificity
- Confusion matrix

		Actual Class	
		Positive	Negative
Test outcome	Predicted positive	**True positive** (TP)	**False positive** (FP, Type I error)
	Predicted negative	**False negative** (FN, Type II error)	**True negative** (TN)
		Sensitivity = $\frac{\#\mathrm{TP}}{\#\mathrm{TP} + \#\mathrm{FN}}$	Specificity = $\frac{\#\mathrm{TN}}{\#\mathrm{TN} + \#\mathrm{FP}}$
		↑	↑
		y-axis	$(1 - x)$-axis

Constructing an ROC Curve

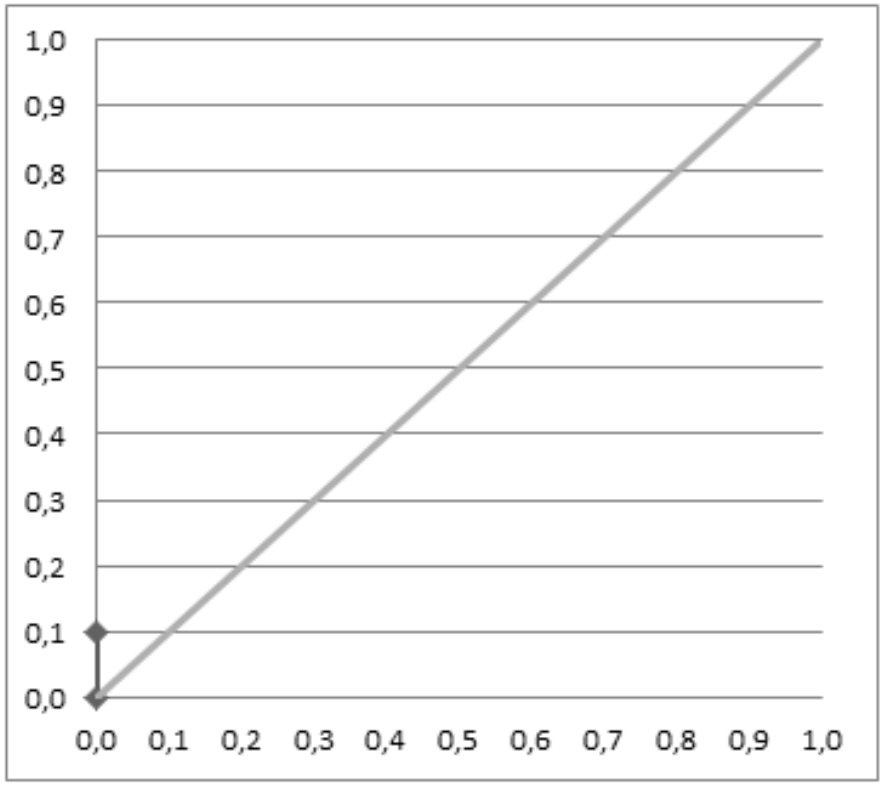
- Example: 20 samples
- True labels (C): 10 positives, and 10 negatives
- Classification rule:

  predicted positive if score >= threshold
- Practical algorithm:
  - Rank samples on decreasing score
  - Start from (0,0)
  - For each sample (in the decreasing order)
    - If C is positive, move 1/pos(=0.1) up
    - If C is negative, move 1/neg(=0.1) right
- E.g. (from top to bottom)
  - Threshold at 0.9, TPR=0.1, FPR=0
  - Threshold at 0.8, TPR=0.2, FPR=0
  - Threshold at 0.7, TPR=0.2, FPR=0.1
  - Threshold at 0.6, TPR=0.3, FPR=0.1
  - Threshold at 0.55, TPR=0.4, FPR=0.1
  - …
- **Only rank ordering matters for ROC analysis!**

#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1

Constructing an ROC Curve

#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1

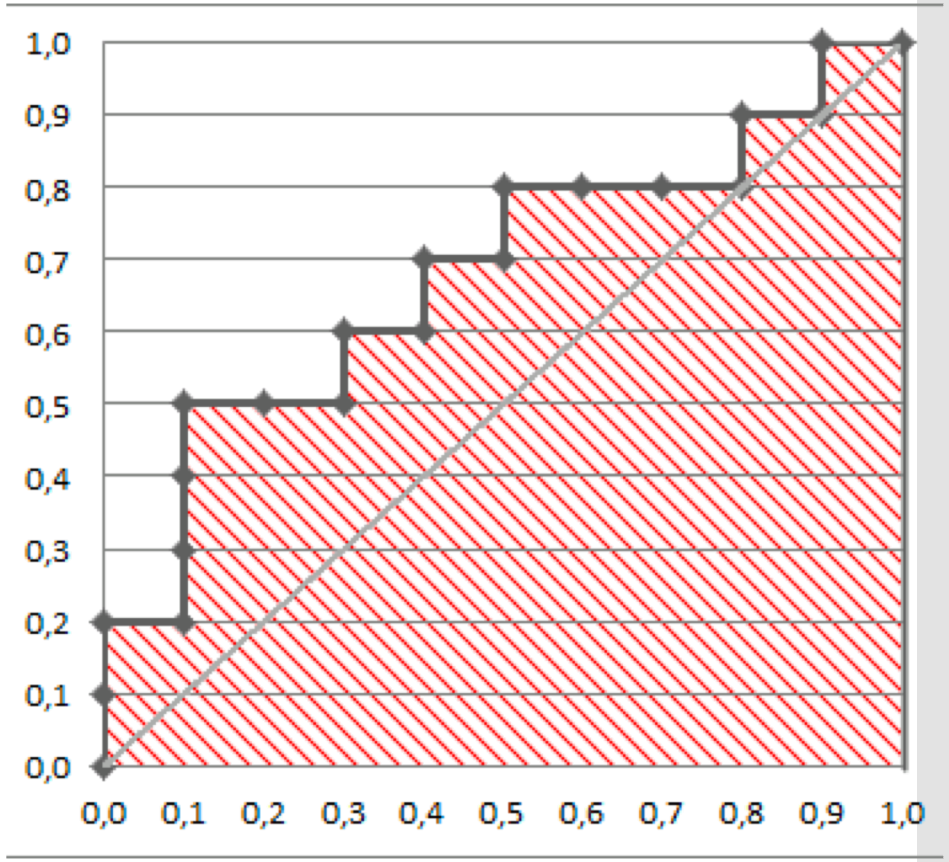


http://mlwiki.org/index.php/ROC_Analysis

Area Under the Curve (AUC)

AUC

- AUC is one single measure that summarizes an ROC curve
- It is the area under the ROC curve
- AUC ranges from 0 to 1
  - AUC = 1: a perfect classifier for which all positives come after all negatives
  - AUC = 0.5: randomly ordered
  - AUC = 0: all negatives come before all positives



Evaluating ROC in R

```
library(ROCR)
prediction(predictions, labels, label.ordering = NULL)
performance(prediction.obj, measure, x.measure="cutoff", ...)
```

Examples

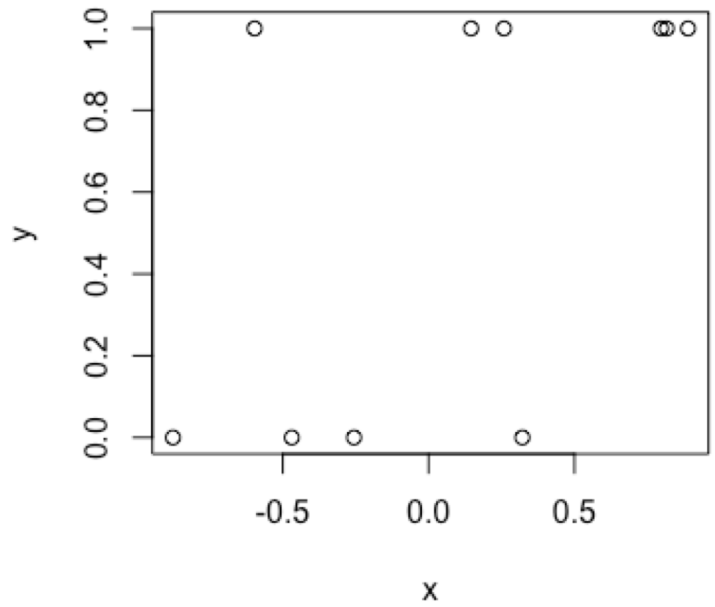
Example I: Logistic Regression

- Suppose we want to simulate 10 samples from the following logistic regression model

$$Y \sim \text{Bernoulli}(p)$$
$$\text{logit}(p) = \beta_0 + \beta_1 x$$

- Set $\beta_0 = 0.6$ and $\beta_1 = 2.8$
- Suppose x is a continuous covariate uniformly distributed from -1 to 1
- We will use `runif()` for generating x, and `rbinom()` for y

```
> ## simulate data
> set.seed(1)
> nsamp <- 10
> x <- runif(nsamp, -1, 1)
> logit.p <- 0.6 + 2.8*x
> p <- inv.logit(logit.p)
> y <- rbinom(nsamp, 1, p)
> plot(x,y)
```



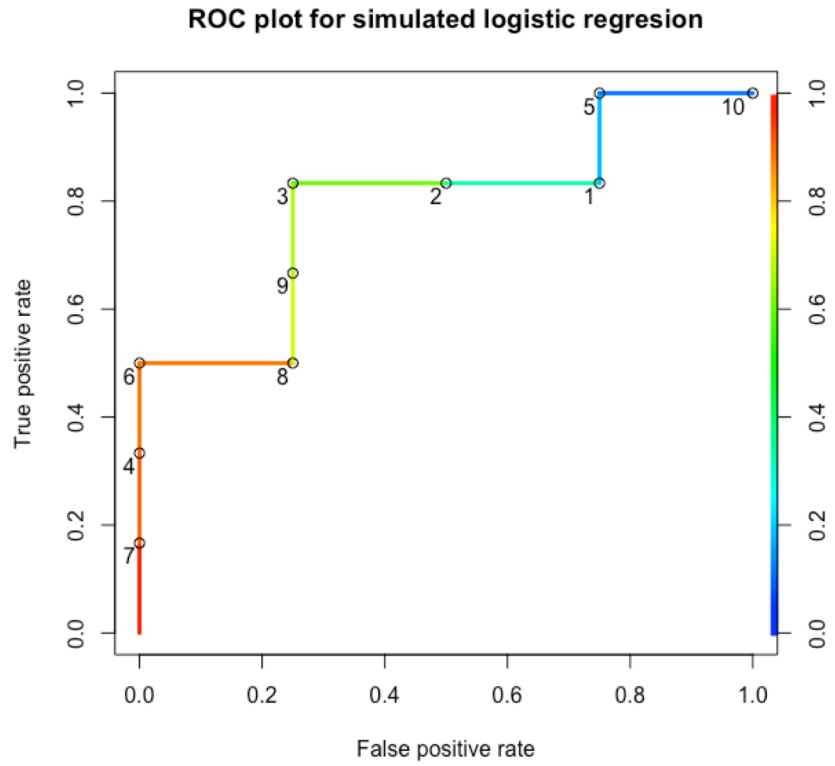
Example I: Logistic Regression

- Fit a generalized linear model and find the fitted values (i.e. predicted scores)
- The true binary values are the y's

```
> ## fit glm and find predicted scores
> mod <- glm(y~x, family=binomial(link = "logit"))
> predicted <- mod$fitted.values
> cbind(predicted, y)[order(predicted, decreasing = T),] #decreasing order
   predicted y
7  0.9268960 1
4  0.9136225 1
6  0.9096926 1
8  0.7556338 0
9  0.7253981 1
3  0.6663547 1
2  0.4240859 0
1  0.3023832 0
5  0.2398980 1
10 0.1360352 0
```

Example I: Logistic Regression

```
> ## ROC plot
> pred <- prediction(predicted, y)
> perf <- performance(pred, "tpr", "fpr")
> plot(perf, lwd=3, colorize=TRUE,
+      print.cutoffs.at=predicted,
+      cutoff.label.function=function(x){NULL},
+      text.adj=c(1.3,1.3),
+      main="ROC plot for simulated logistic regresion")
```



	predicted	y
7	0.9268960	1
4	0.9136225	1
6	0.9096926	1
8	0.7556338	0
9	0.7253981	1
3	0.6663547	1
2	0.4240859	0
1	0.3023832	0
5	0.2398980	1
10	0.1360352	0

Example II: Hypothesis Testing

- Consider a two-group comparison problem

$$H_0: \mu_X = \mu_Y \text{ versus } H_1: \mu_X \neq \mu_Y$$

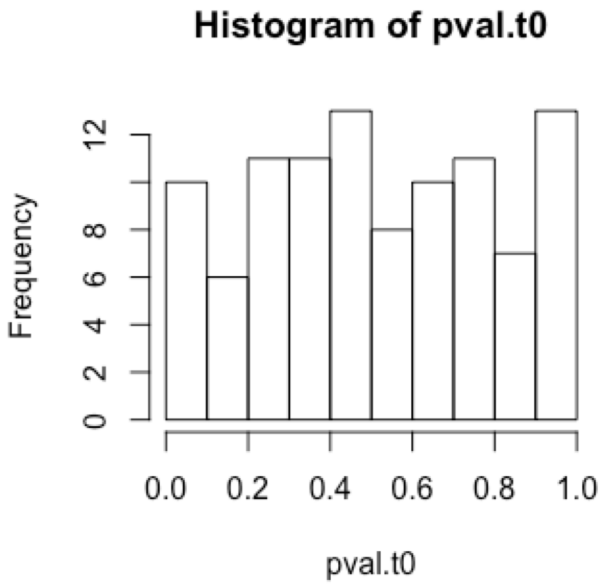
- Data generation
  - Under the null, suppose $X \sim N(0,1)$ and $Y \sim N(0,1)$
  - Under the alternative, suppose $X \sim N(0,1)$ and $Y \sim N(0.5,1)$
  - Set sample size $n = 30$ per group
  - Repeat null and alternative "experiments" 100 times for each

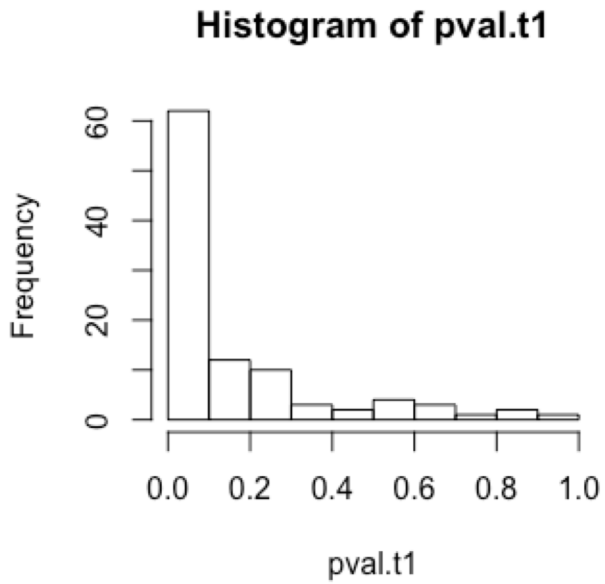
```
> ## data generation
> set.seed(123)
> n <- 30
> Niter <- 100
>
> XX0 <- matrix(rnorm(n*Niter, 0, 1), nrow=n)
> YY0 <- matrix(rnorm(n*Niter, 0, 1), nrow=n)
> XX1 <- matrix(rnorm(n*Niter, 0, 1), nrow=n)
> YY1 <- matrix(rnorm(n*Niter, 0.5, 1), nrow=n)
> Grp <- as.factor(rep(c("X","Y"), each=n)) #two groups
>
> mydat0 <- rbind(XX0,YY0) #null data
> mydat1 <- rbind(XX1,YY1) #alternative data
```

Example II: Hypothesis Testing

- Perform t-test and look at the p-values

```
> ## t-test
> library(genefilter)
> pval.t0 <- colttests(mydat0, Grp)[,"p.value"]
> hist(pval.t0)
> pval.t1 <- colttests(mydat1, Grp)[,"p.value"]
> hist(pval.t1)
> pval.t <- c(pval.t0, pval.t1)
```

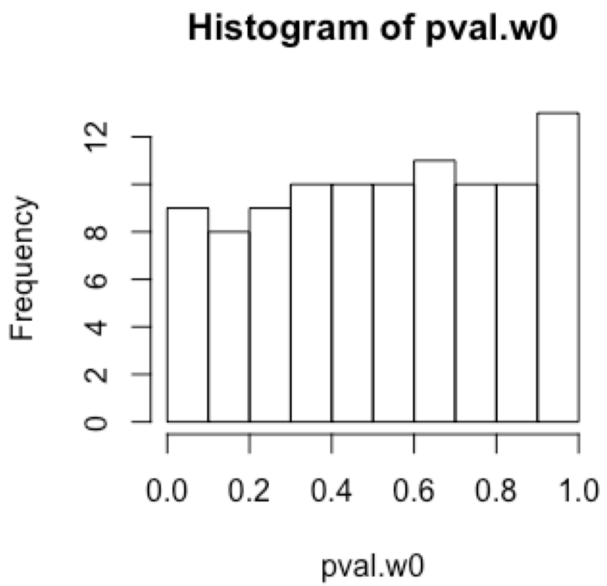


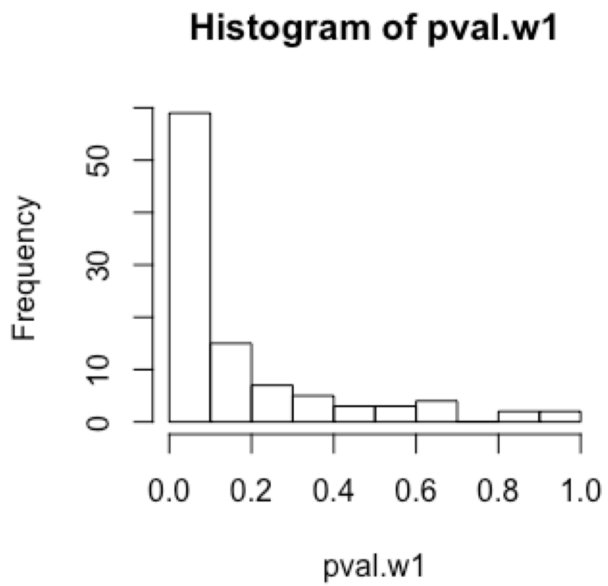


Example II: Hypothesis Testing

- Perform Wilcoxon test and look at the p-values

```
> ## Wilcoxon test
> pval.w0 <- apply(mydat0, 2, function(z) wilcox.test(z~Grp)$p.value)
> hist(pval.w0)
> pval.w1 <- apply(mydat1, 2, function(z) wilcox.test(z~Grp)$p.value)
> hist(pval.w1)
> pval.w <- c(pval.w0, pval.w1)
```



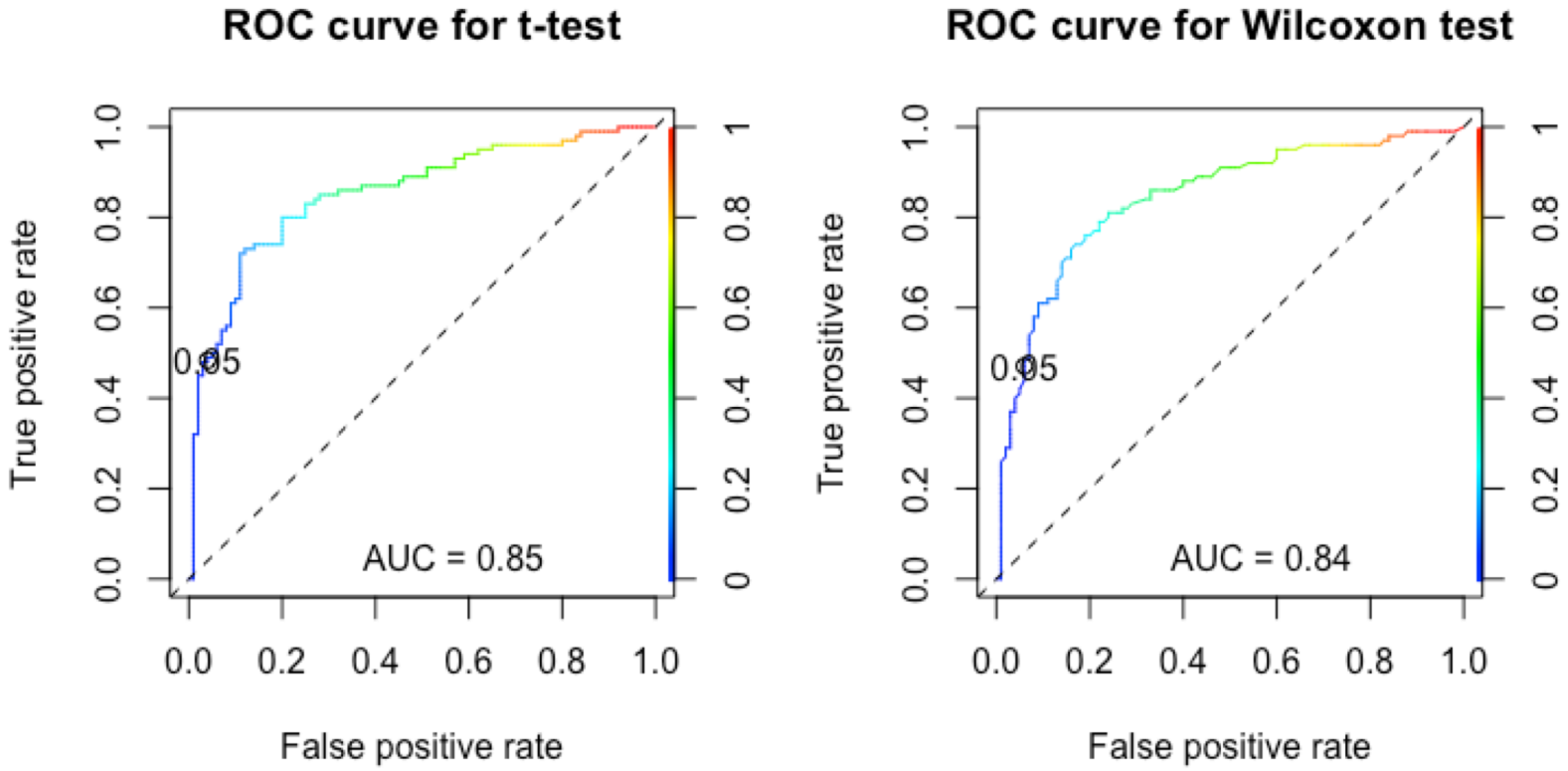


Example II: Hypothesis Testing

- Plotting ROC curve with p-values is tricky!
- Since null data are expected to have larger (>=0.05) p-values, and alternative data are expected to have smaller (<0.05) p-values
- Set the true labels: null = 1, alternative = 0
- Calculate the "tnr" and "fnr" measures in the performance() function
- Don't forget to manually specify the axis labels for the plot

```
> library(ROCR)
> mylab <- rep(1:0, each=Niter) #set true labels
>
> pred.t <- prediction(pval.t, mylab)
> perf.t <- performance(pred.t, "tnr", "fnr")
> auc.t <- performance(pred.t, "auc")@y.values[[1]] #get AUC
> plot(perf.t, main="ROC curve for t-test",
+      xlab="False positive rate", ylab="True positive rate",
+      colorize=TRUE, print.cutoffs.at=c(0.05))
> legend("bottom", paste0("AUC = ", round(auc.t,2)), bty = "n")
> abline(0,1,lty=2)
```

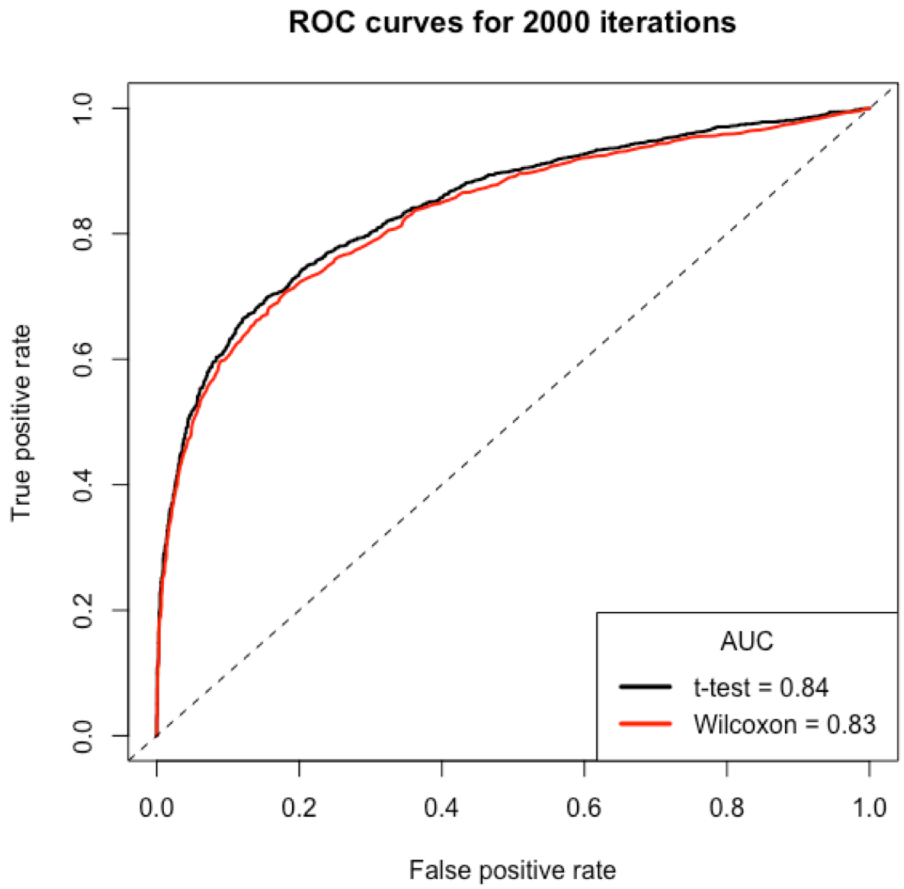
Example II: Hypothesis Testing



- Also, x-axis is the type-I error rate, and y-axis is the power of a hypothesis test

Example II: Hypothesis Testing

- Overlay ROC curves show that the t-test (parametric) is *uniformly better* than the Wilcoxon test (non-parametric), since our data are generated from Normal distributions



Understanding ROC curves

- http://www.navan.name/roc/