Section 8 : Matching III

Andrew Bertoli

22 October 2013

Roadmap

1. KS Tests
2. Genetic Matching
3. General Questions
4. Homework Questions

KS Tests

The KS test is a non-parametric test of whether two distributions are equal.

There are two types of KS Tests:

1. A one sample test that compares an empirical distribution to a known reference distribution

2. A two sample test that compares the empirical distributions of two samples

The null in the one sample test is that the sample was drawn from the reference distribution, and the null in the two sample test is that the two samples were drawn from the same distribution.

KS Tests

The Empirical CDF

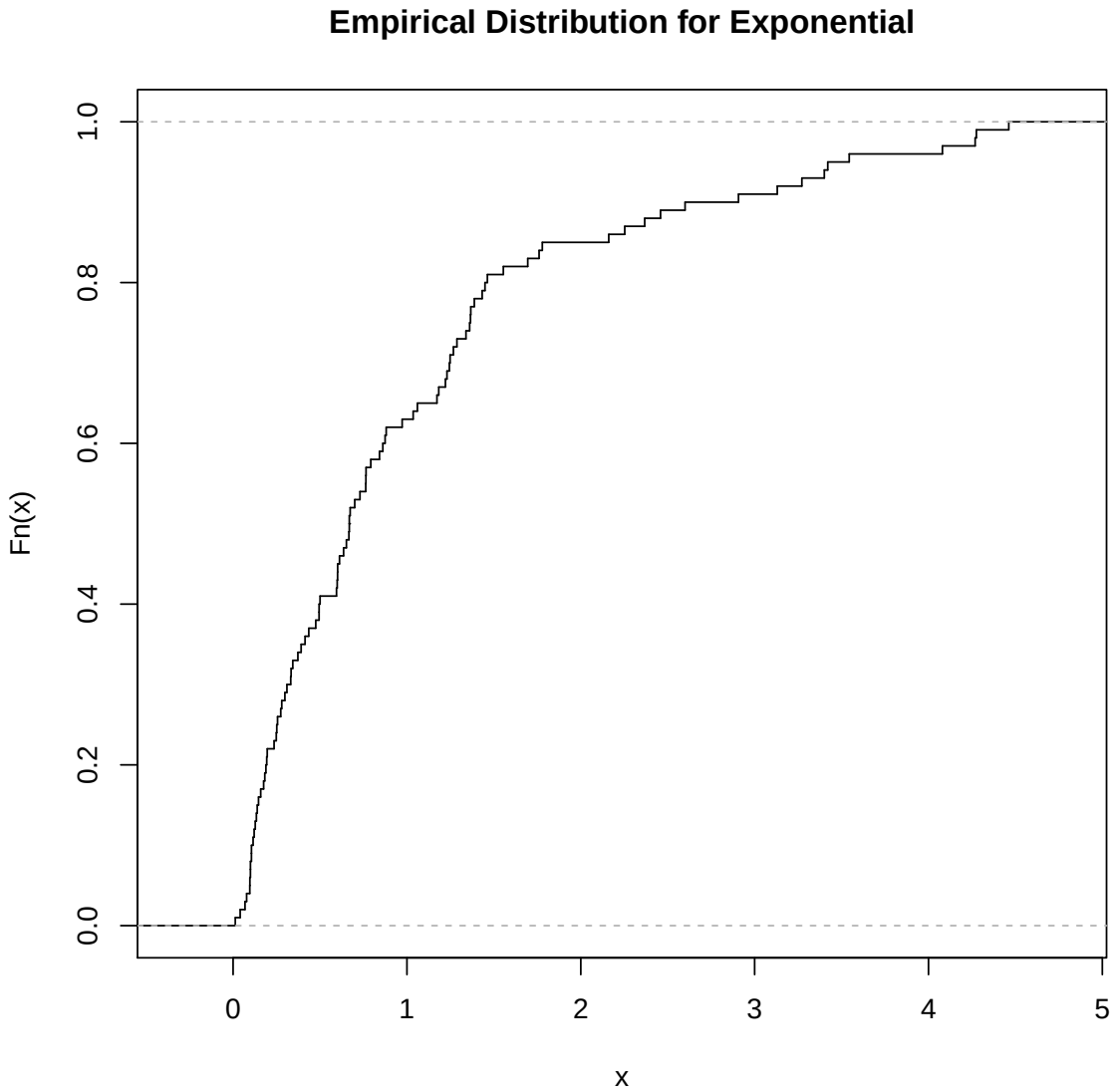
For a sample of i.i.d. random variables, the empirical distribution is

$$\hat{F}_X(x) = \frac{\sum_{i=1}^{n} I(X_i \leq x)}{n}$$

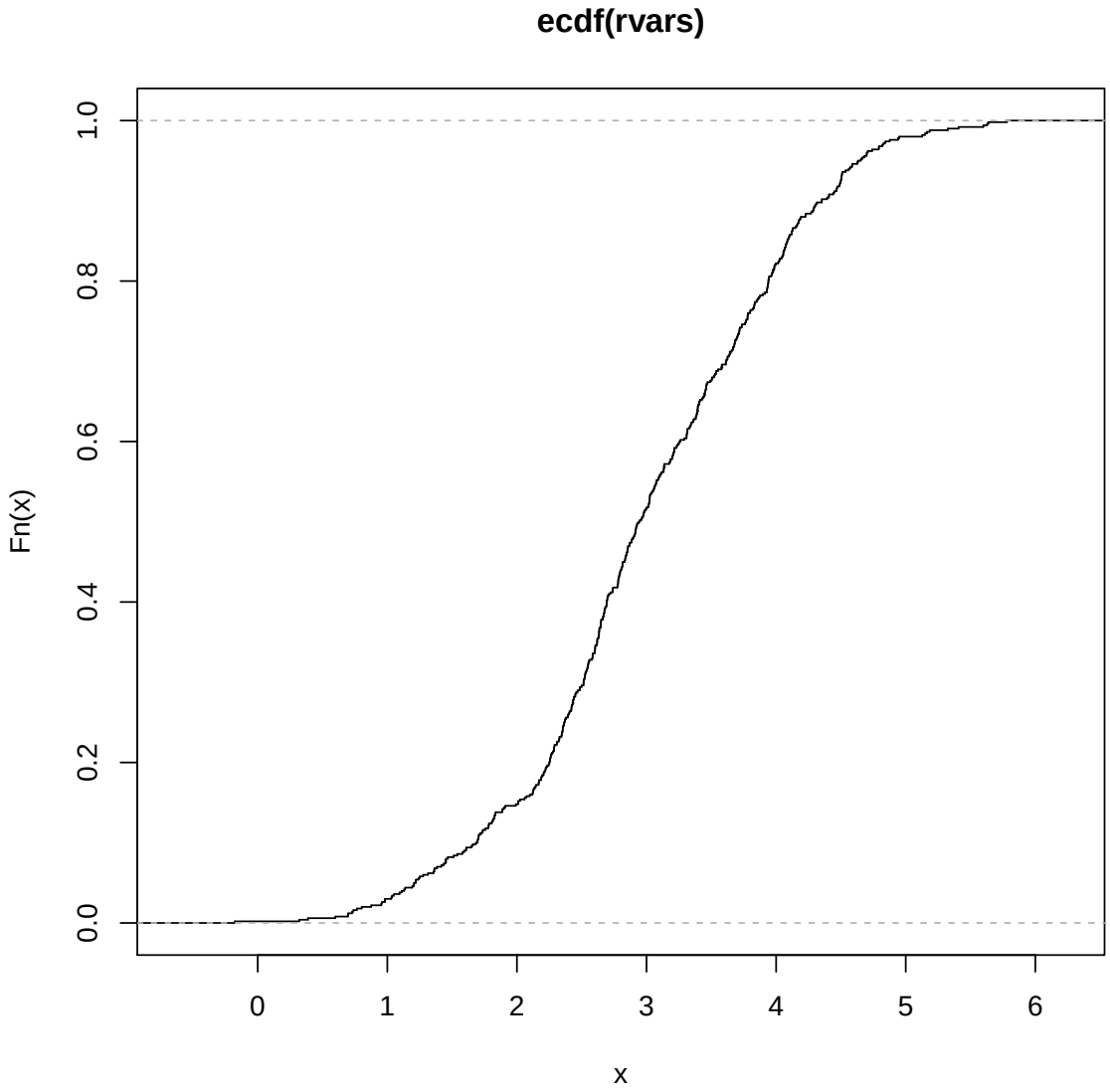
where I is the indicator function.

In other words, the value of the empirical distribution at any x is equal to the proportion of points in the sample that are less than or equal to x.

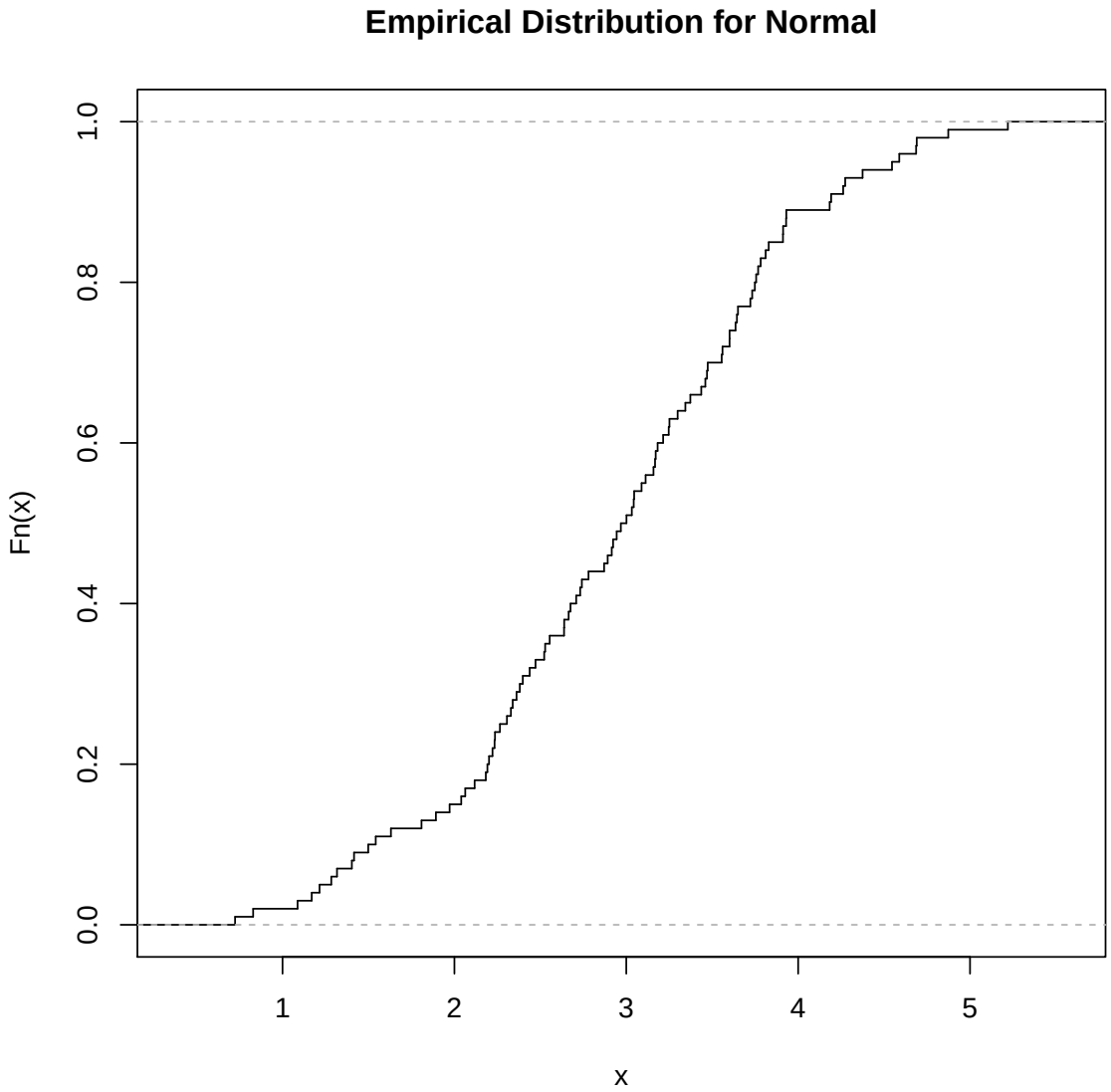
KS Tests


Empirical Distribution for Exponential
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
x

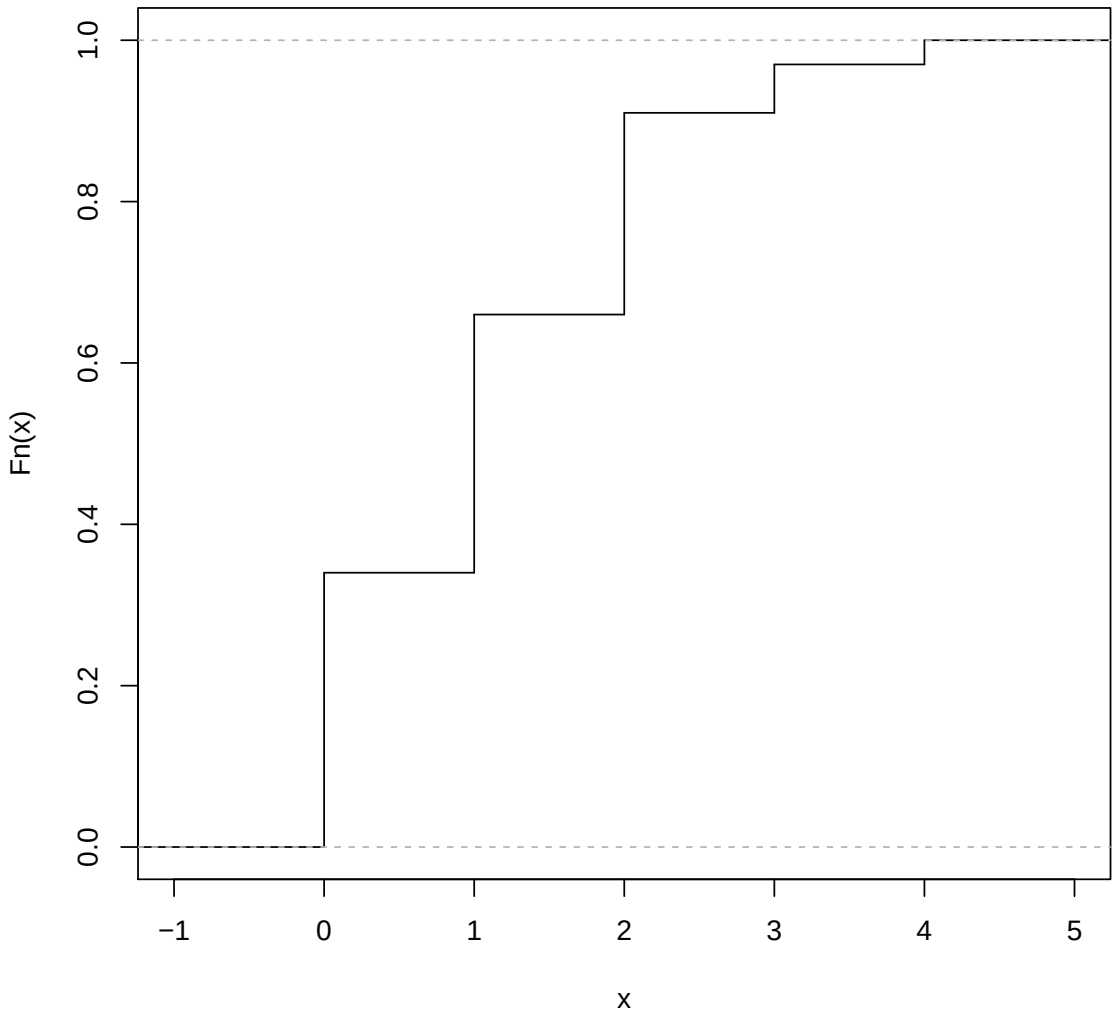
KS Tests



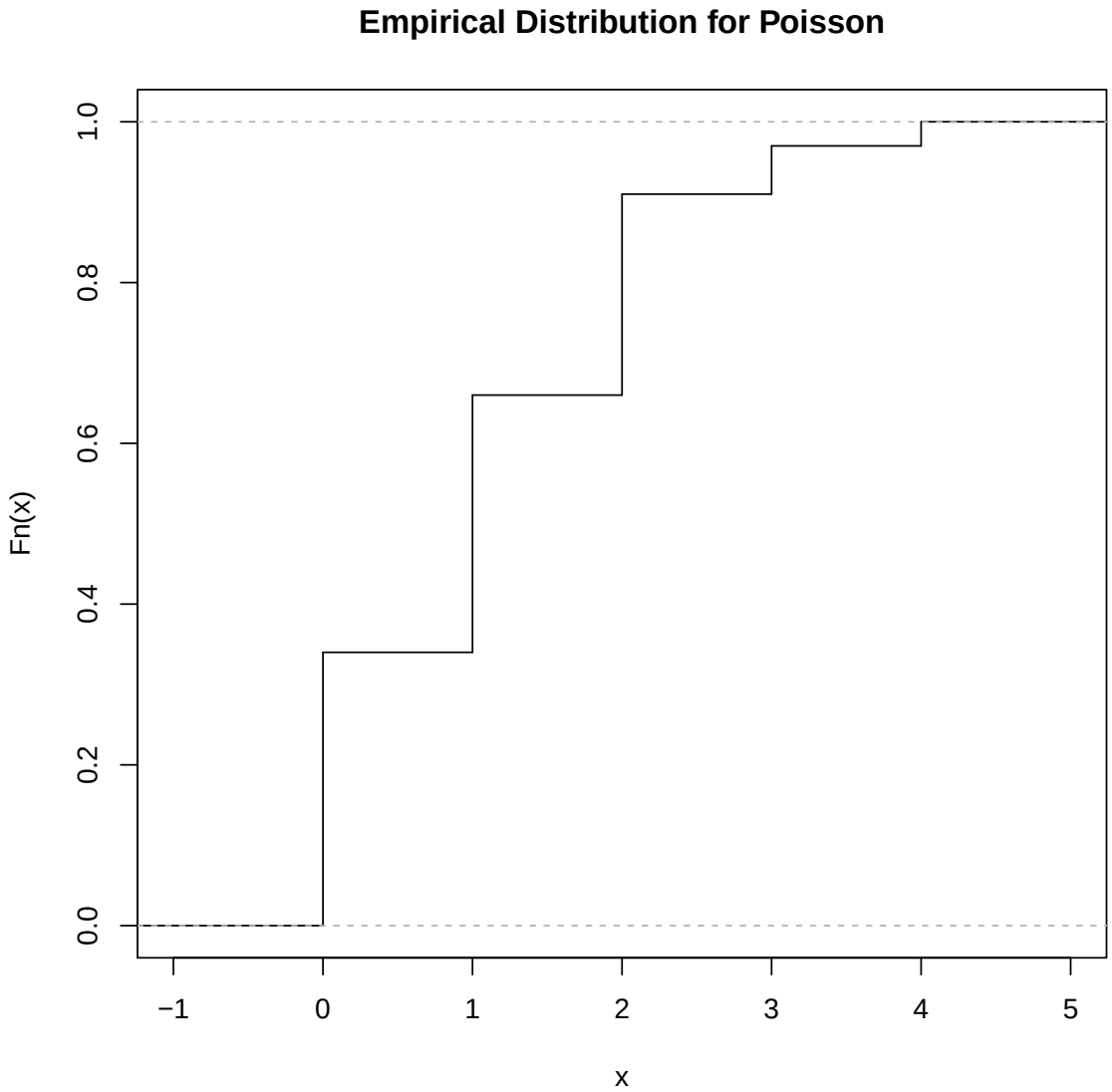
KS Tests


Empirical Distribution for Normal
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5
x

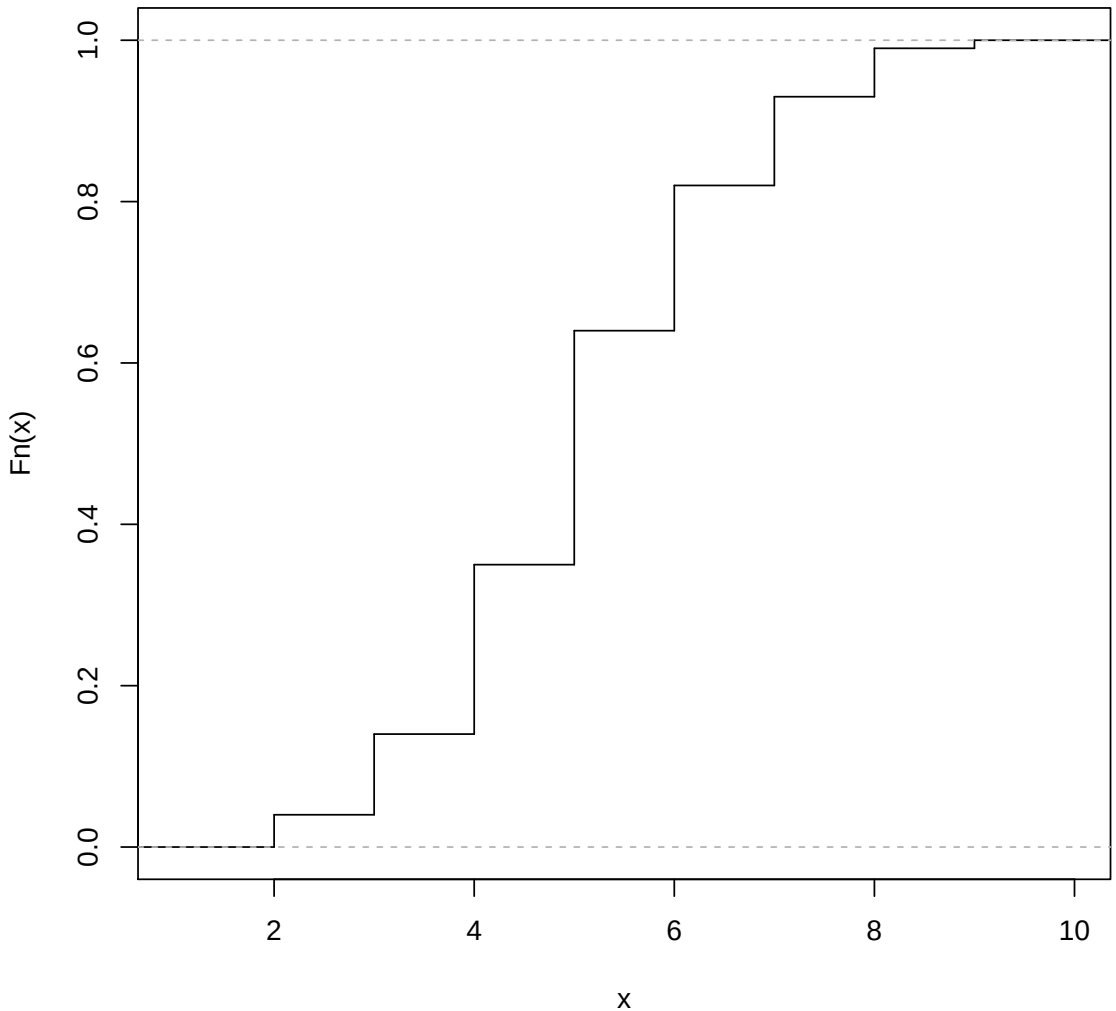
KS Tests



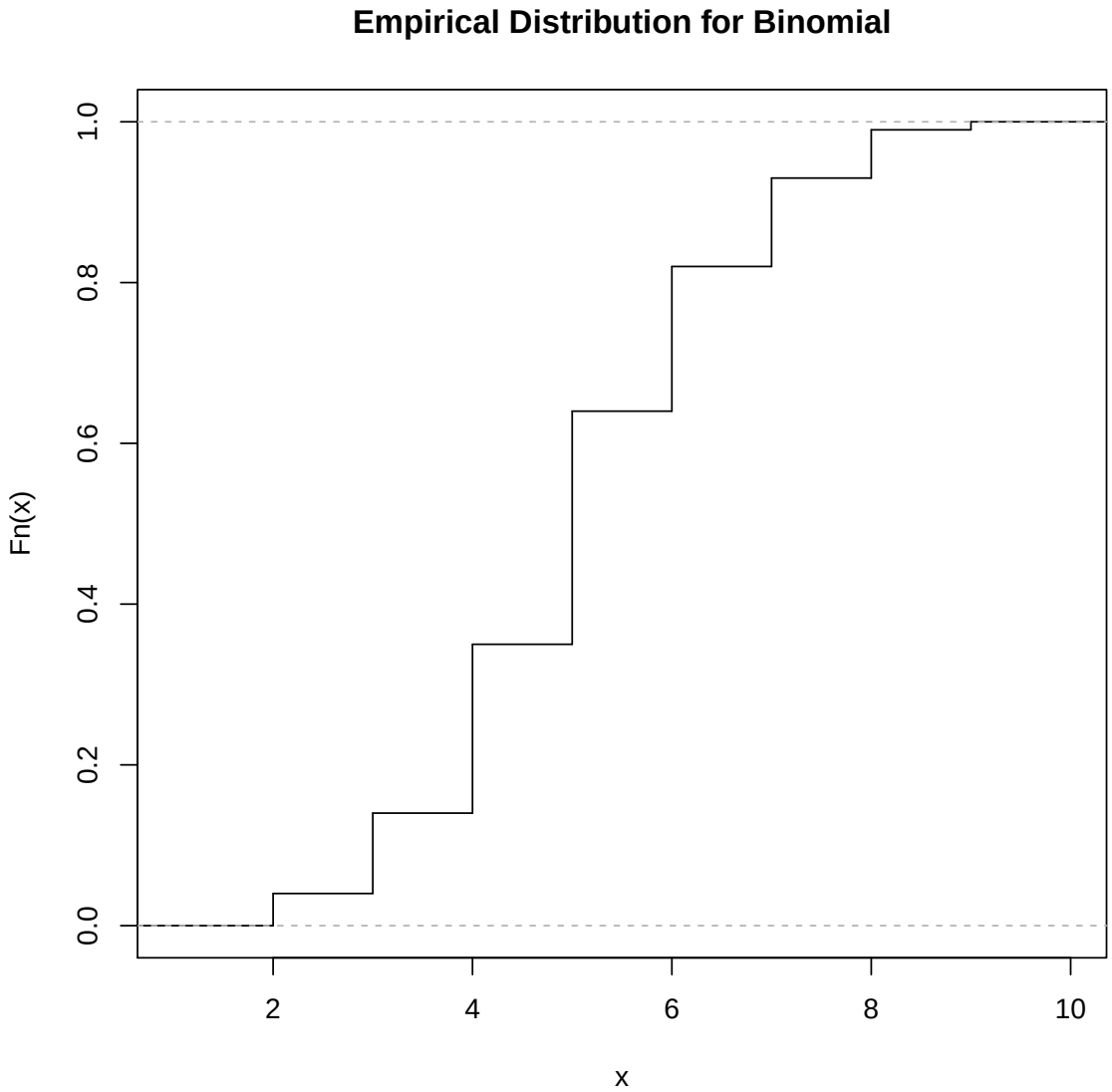
KS Tests


Empirical Distribution for Poisson
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
−1
0
1
2
3
4
5
x

KS Tests



KS Tests


Empirical Distribution for Binomial
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
x

KS Tests

For a one sample test, the test statistic is the maximum vertical difference between the empirical CDF and the reference CDF. (You will normally need to normalize your sample distribution.)

For a two sample test, the test statistic is the maximum vertical difference between the two empirical CDFs.

This value will go to 0 as $n \to \infty$

KS Tests

One Sample Example

```
> sample=rnorm(n=100, mean=3, sd=1)
>
> normalized.sample=(sample-mean(sample))/sd(sample)
>
> ks.test(x=normalized.sample, y=pnorm)

    One-sample Kolmogorov-Smirnov test

data:  normalized.sample
D = 0.057, p-value = 0.9014
alternative hypothesis: two-sided
```

KS Tests

Two Sample Example

```
> sample1=rnorm(n=100, mean=3, sd=1)
>
> sample2=rnorm(n=100, mean=3, sd=2)
>
> ks.test(sample1, sample2)

    Two-sample Kolmogorov-Smirnov test

data:  sample1 and sample2
D = 0.22, p-value = 0.01581
alternative hypothesis: two-sided
```

KS Tests

The t-test fails to reject the null.

```
> t.test(sample1, sample2)

    Welch Two Sample t-test

data:  sample1 and sample2
t = -0.8438, df = 149.622, p-value = 0.4001
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.6378883  0.2561223
sample estimates:
mean of x mean of y
 2.978541  3.169424
```

Genetic Matching

Mahalanobis distance matching picks matches that minimize this distance metric.

$$md(\mathbf{X_i}, \mathbf{X_j}) = [(\mathbf{X_i} - \mathbf{X_j})^\top \mathbf{S}^{-1} (\mathbf{X_i} - \mathbf{X_j})]^{1/2}$$

Genetic matching matches using this distract metric:

$$d(\mathbf{X_i}, \mathbf{X_j}) = [(\mathbf{X_i} - \mathbf{X_j})^\top (\mathbf{S}^{-1/2})' \mathbf{W} \mathbf{S}^{-1/2} (\mathbf{X_i} - \mathbf{X_j})]^{1/2}$$

and searches for the diagonal matrix $\mathbf{W}$ that minimizes covariate imbalance.

Genetic Matching

Load the package 'Matching'

Matching includes the functions GenMatch(), Matching(), and MatchBalance().

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1,
weights=NULL, pop.size = 100, max.generations=100,
wait.generations=4, hard.generation.limit=FALSE,
starting.values=rep(1,ncol(X)), fit.func="pvals",
MemoryMatrix=TRUE, exact=NULL, caliper=NULL,
replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0,
ks=TRUE, verbose=FALSE, distance.tolerance=1e-05,
tolerance=sqrt(.Machine$double.eps), min.weight=0,
max.weight=1000, Domains=NULL, print.level=2,
project.path=NULL, paired=TRUE, loss=1,
data.type.integer=FALSE, restrict=NULL, cluster=FALSE,
balance=TRUE, ...)

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1,
weights=NULL, pop.size = 100, max.generations=100,
wait.generations=4, hard.generation.limit=FALSE,
starting.values=rep(1,ncol(X)), fit.func="pvals",
MemoryMatrix=TRUE, exact=NULL, caliper=NULL,
replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0,
ks=TRUE, verbose=FALSE, distance.tolerance=1e-05,
tolerance=sqrt(.Machine$double.eps), min.weight=0,
max.weight=1000, Domains=NULL, print.level=2,
project.path=NULL, paired=TRUE, loss=1,
data.type.integer=FALSE, restrict=NULL, cluster=FALSE,
balance=TRUE, ...)

Genetic Matching

Use GenMatch() to find the matches

GenMatch(**Tr**, X, BalanceMatrix=X, estimand="ATT", M=1,
weights=NULL, pop.size = 100, max.generations=100,
wait.generations=4, hard.generation.limit=FALSE,
starting.values=rep(1,ncol(X)), fit.func="pvals",
MemoryMatrix=TRUE, exact=NULL, caliper=NULL,
replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0,
ks=TRUE, verbose=FALSE, distance.tolerance=1e-05,
tolerance=sqrt(.Machine$double.eps), min.weight=0,
max.weight=1000, Domains=NULL, print.level=2,
project.path=NULL, paired=TRUE, loss=1,
data.type.integer=FALSE, restrict=NULL, cluster=FALSE,
balance=TRUE, ...)

Tr is the vector of treatment assignments.

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, **X**, BalanceMatrix=X, estimand="ATT", M=1,
weights=NULL, pop.size = 100, max.generations=100,
wait.generations=4, hard.generation.limit=FALSE,
starting.values=rep(1,ncol(X)), fit.func="pvals",
MemoryMatrix=TRUE, exact=NULL, caliper=NULL,
replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0,
ks=TRUE, verbose=FALSE, distance.tolerance=1e-05,
tolerance=sqrt(.Machine$double.eps), min.weight=0,
max.weight=1000, Domains=NULL, print.level=2,
project.path=NULL, paired=TRUE, loss=1,
data.type.integer=FALSE, restrict=NULL, cluster=FALSE,
balance=TRUE, ...)

X is the vector of variables that we want to match on.

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, **BalanceMatrix**=X, estimand="ATT", M=1, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.func="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

BalanceMatrix is the vector of variables that we wish to achieve balance on (defaults to **X**).

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, **estimand**="ATT", M=1, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.func="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

estimand is the parameter of interest. It can be the "ATT", "ATE", or "ATC".

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", **M = 1**, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.func="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

M is the number of control units that you match to each treated unit. M=1 means one-to-one matching.

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1, **weights**=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.func="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

weights is a vector with the same length as Y that specifies the weights to be placed on the observations. This is **not** the weights denoting the importance of the covariates.

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.fun="pvals", MemoryMatrix=TRUE, exact=NULL, **caliper**=NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

caliper is a vector denoting the maximum difference apart each match can be for every covariate. Units with no suitable matches will be dropped.

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.fun="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, **replace**=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

replace allows you to choose whether to match with or without replacement.

Genetic Matching

Use GenMatch() to find the matches

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.fun="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, replace=TRUE, **ties**=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, loss=1, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

ties allows you to deal with cases where there are several equally suitable control units for one treated unit. If **ties** is set at TRUE, GenMatch will average over the control units. If **ties** is FALSE, ties will be broken with a coin flip.

Genetic Matching

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1,
weights=NULL, pop.size = 100, max.generations=100,
wait.generations=4, hard.generation.limit=FALSE,
starting.values=rep(1,ncol(X)), fit.fun="pvals",
MemoryMatrix=TRUE, exact=NULL, caliper=NULL,
replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0,
ks=TRUE, verbose=FALSE, distance.tolerance=1e-05,
tolerance=sqrt(.Machine$double.eps), min.weight=0,
max.weight=1000, Domains=NULL, print.level=2,
project.path=NULL, **paired**=TRUE, loss=1,
data.type.integer=FALSE, restrict=NULL, cluster=FALSE,
balance=TRUE, ...)

paired allows you to choose whether GenMatch uses paired t-tests.

Genetic Matching

GenMatch(Tr, X, BalanceMatrix=X, estimand="ATT", M=1, weights=NULL, pop.size = 100, max.generations=100, wait.generations=4, hard.generation.limit=FALSE, starting.values=rep(1,ncol(X)), fit.fun="pvals", MemoryMatrix=TRUE, exact=NULL, caliper=NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, nboots=0, ks=TRUE, verbose=FALSE, distance.tolerance=1e-05, tolerance=sqrt(.Machine$double.eps), min.weight=0, max.weight=1000, Domains=NULL, print.level=2, project.path=NULL, paired=TRUE, **loss = 1**, data.type.integer=FALSE, restrict=NULL, cluster=FALSE, balance=TRUE, ...)

loss is the loss function. It should take the vector of p-values for the covariates and return some value that GenMatch will try to maximize. The default is to sort the p-values and minimize the maximum discrepancy.

Genetic Matching

Match(**Y**=NULL, Tr, X, Z = X, V = rep(1, length(Y)), estimand = "ATT", M = 1, BiasAdjust = FALSE, exact = NULL, caliper = NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE,Weight = 1, Weight.matrix = NULL, weights = NULL, Var.calc = 0, sample = FALSE, restrict=NULL, match.out = NULL, distance.tolerance = 1e-05, tolerance=sqrt(.Machine$double.eps), version="standard")

Y is the vector of outcomes.

Genetic Matching

Match(Y=NULL, **Tr**, X, Z = X, V = rep(1, length(Y)), estimand = "ATT", M = 1, BiasAdjust = FALSE, exact = NULL, caliper = NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE,Weight = 1, Weight.matrix = NULL, weights = NULL, Var.calc = 0, sample = FALSE, restrict=NULL, match.out = NULL, distance.tolerance = 1e-05, tolerance=sqrt(.Machine$double.eps), version="standard")

Tr is the treatment assignment vector.

Genetic Matching

Match(Y=NULL, Tr, **X**, Z = X, V = rep(1, length(Y)), estimand = "ATT", M = 1, BiasAdjust = FALSE, exact = NULL, caliper = NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE,Weight = 1, Weight.matrix = NULL, weights = NULL, Var.calc = 0, sample = FALSE, restrict=NULL, match.out = NULL, distance.tolerance = 1e-05, tolerance=sqrt(.Machine$double.eps), version="standard")

X is the matrix of control variables.

Genetic Matching

Match(Y=NULL, Tr, X, Z = X, V = rep(1, length(Y)), estimand = "ATT", M = 1, BiasAdjust = FALSE, exact = NULL, caliper = NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE, **Weight** = 1, Weight.matrix = NULL, weights = NULL, Var.calc = 0, sample = FALSE, restrict=NULL, match.out = NULL, distance.tolerance = 1e-05, tolerance=sqrt(.Machine$double.eps), version="standard")

Weight can be set at 2 to do Mahalanobis Distance matching. Leave this argument blank if you want to do Genetic matching.

Genetic Matching

Match(Y=NULL, Tr, X, Z = X, V = rep(1, length(Y)), estimand = "ATT", M = 1, BiasAdjust = FALSE, exact = NULL, caliper = NULL, replace=TRUE, ties=TRUE, CommonSupport=FALSE,Weight = 1, **Weight.matrix** = NULL, weights = NULL, Var.calc = 0, sample = FALSE, restrict=NULL, match.out = NULL, distance.tolerance = 1e-05, tolerance=sqrt(.Machine$double.eps), version="standard")

Weight.matrix takes the output from the GenMatch() function.

Genetic Matching

```
> library(Matching)
gen=GenMatch(Tr=data$etouch,X=with(data,cbind(income, b2000, hispanic, size)))

Match(Y=data$b2004, Tr=data$etouch, X=with(data,cbind(income, b2000, hispanic, size)), Weight.matrix=gen)
```

Genetic Matching

```
$est
      [,1]
[1,] 20933

$se
[1] 28768.51

$est.noadj
[1] 20933

$se.standard
[1] 23754.81
```

Genetic Matching

$est is the estimated treatment effect (adjusted for bias if BiasAdjust=TRUE)

$se is the estimated standard error (adjusted for the uncertainty in the matching procedure)

$est.noadj is the estimated treatment effect (not adjusted for bias)

$se.standard is the standard error estimated the normal way

Genetic Matching

```
$index.treated
 [1]  6  8 11 28 30 34 35 42 43 45 50 51 52 56 60

$index.control
 [1] 48  3 48 48 17 36 40 46 25 10 48 64 15 64 19
```

Genetic Matching

MatchBalance(formul, data = NULL, match.out = NULL, ks = TRUE, nboots=500, weights=NULL, digits=5, paired=TRUE, print.level=1)

Genetic Matching

MatchBalance(**formul**, data = NULL, match.out = NULL, ks = TRUE, nboots=500, weights=NULL, digits=5, paired=TRUE, print.level=1)

formul is not the regular formula. It should be written as Treat $\sim$ Control 1 + Control 2 +...

Genetic Matching

MatchBalance(formul, **data** = NULL, match.out = NULL, ks = TRUE, nboots=500, weights=NULL, digits=5, paired=TRUE, print.level=1)

data is the data frame.

Genetic Matching

MatchBalance(formul, data = NULL, **match.out** = NULL, ks = TRUE, nboots=500, weights=NULL, digits=5, paired=TRUE, print.level=1)

match.out is the output of Match(). If you do not include this, MatchBalance will only return the balance before matching.

Genetic Matching

MatchBalance(formul, data = NULL, match.out = NULL, ks = TRUE, nboots=500, weights=NULL, digits=5, **paired**=TRUE, print.level=1)

paired determines if the t.tests on the matched data are paired.

Genetic Matching

```
> mat=Match(Y=data$b2004, Tr=data$etouch, X=with(data,cbind(income, b2000, hispanic,
size)), Weight.matrix=gen)
>
> MatchBalance(etouch ~ income + b2000 + hispanic + size, data=data, match.out=mat)
```

Genetic Matching

```
***** (V1) income *****
                          Before Matching     After Matching
mean treatment........      39282              39282
mean control..........      34261              39358
std mean diff.........     105.39             -1.5994

mean raw eQQ diff.....     5574.9              1128.3
med  raw eQQ diff.....       6222                1045
max  raw eQQ diff.....       7100                4158

mean eCDF diff........    0.26883            0.066667
med  eCDF diff........    0.25513            0.066667
max  eCDF diff........    0.52949                 0.2

var ratio (Tr/Co).....    0.56619              1.0204
T-test p-value........  0.0023728             0.88572
KS Bootstrap p-value.. < 2.22e-16                0.85
KS Naive p-value......  0.0016245             0.92509
KS Statistic..........    0.52949                 0.2
```

Matching

Remember, Genetic Matching solves only one of these problems.

1. There might not be support in the data.

2. There might be support, but you chose the wrong X's.

3. You might have support and the right X's, but your formula for the propensity score is wrong (if you are doing propensity score matching) or your controls do not each follow an elliptic distribution (if you are doing Mahalanobis distance matching).

4. Everything else worked, but there was noise in your controls.

5. Everything worked perfectly, but people will still be skeptical or think that you p-hacked.

Matching

Example of Z-Bias

Say we create a new vitamin called Vitamin X. We want to estimate the effect of taking Vitamin X on health. We have 100 people, so we randomly select 50 of them to have free Vitamin X pills for a year. This way, we are not forcing anyone to take Vitamin X. Twenty-five people accept the offer.

Question: What is the better comparison?

a) The 25 people who took the Vitamin X to the 75 people who did not.

b) The 25 people who took Vitamin X to the 25 people who were offered and did not take it.

Matching

Example of Z-Bias

You can think of the offer for free Vitamin X as an instrument that increased the number of people who took the vitamin.

If we match our treated units to the untreated units that were offered the vitamin, our estimator would probably be more biased.

General Questions

1. Lecture
2. Section
3. Readings

Homework Questions