Continuous Distributions

- pdf $f_X(x)$: `dnorm(x)` for "density"
- cdf $F_X(x) = \int_{-\infty}^{x} f_X(y)dy$: `pnorm(x)` for "probability"
- qf $Q_X(u) = F_X^{-1}(u)$: `qnorm(u)` for "quantile"

For standard normal,

```
> c(dnorm(-1.4),dnorm(1.4))
[1] 0.1497275 0.1497275
> pnorm(1.4)
[1] 0.9192433
> qnorm(.95)
[1] 1.644854
> qnorm(.975)
[1] 1.959964
```

Distribution parameters may be specified:

```
> pnorm(15.5, mean=10, sd=3)
[1] 0.9666235
```

Probabilities (and quantiles) may be for the upper tail:

```
> pnorm(15.5, mean=10, sd=3, lower.tail=FALSE)
[1] 0.03337651
```

Vectors are allowed:

```
> qnorm(c(.05,.25,.5,.75,.95), mean=10, sd=3)
[1]  5.065439  7.976531 10.000000 12.023469 14.934561
>
```


Discrete Distributions

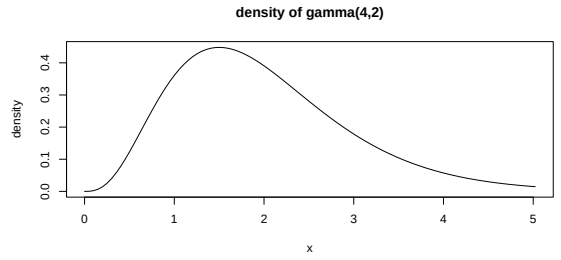
- pmf $f_X(x) = \mathsf{Pr}\{X = x\}$: `dpois(x, lambda)` for "density" (!?)
- cdf $F_X(x) = \sum_{y \le x} \mathsf{Pr}\{X = y\}$: `ppois(x, lambda)` for "probability"
- qf $Q_X(u) = \mathsf{inf}\{x : F_X(x) \ge u\}$: `qpois(u,lambda)` for "quantile"

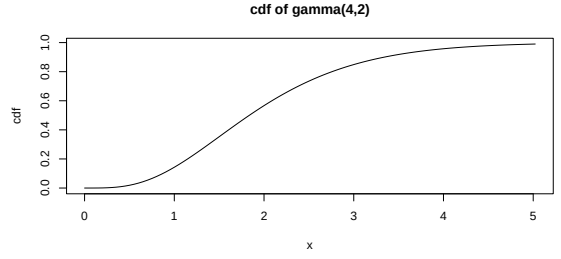
Some examples:

```
> dpois(0,lambda=1)
[1] 0.3678794
> exp(-1)
[1] 0.3678794
> ppois(18,lambda=20)
[1] 0.3814219
> ppois(18,lambda=20,lower.tail=F)
[1] 0.618578
> qpois(c(0,.05,.25,.5,.95,1), lambda=3)
[1]   0   1   2   3   6 Inf
>
```


Plotting Continuous Dists

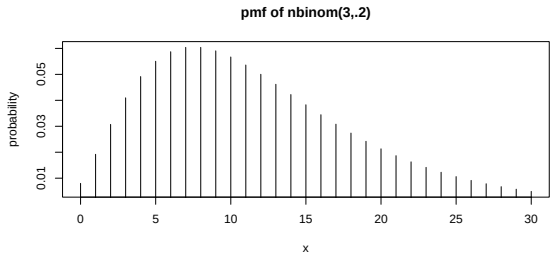
```
> curve(dgamma(x, shape=4, rate=2), 0, qgamma(.99, shape=4, rate=2),
        main="density of gamma(4,2)", ylab="density")
> curve(pgamma(x, shape=4, rate=2), 0, qgamma(.99, shape=4, rate=2),
        main="cdf of gamma(4,2)", ylab="cdf")
>
```

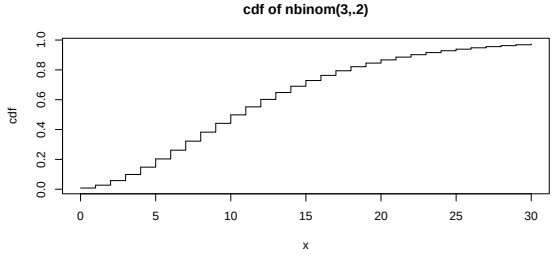




Plotting Discrete Dists

```
> x <- 0:30
> plot(x, dnbinom(x, size=3, prob=.2), type="h",
       main="pmf of nbinom(3,.2)", ylab="probability")
> plot(x, pnbinom(x, size=3, prob=.2), type="s",
       main="cdf of nbinom(3,.2)", ylab="cdf")
>
```





Distributions

Continuous:

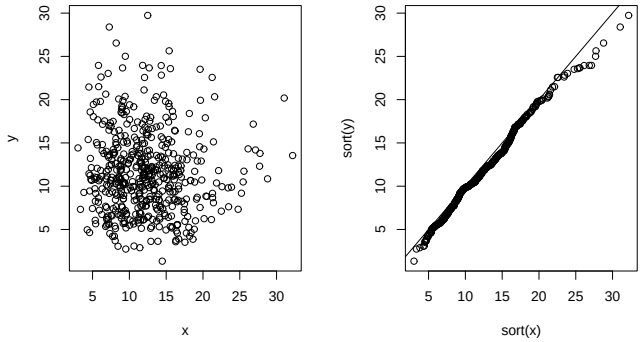
Beta	`beta`	`shape1, shape2`
Cauchy	`cauchy`	`location=0, scale=1`
Chi-Squared	`chisq`	`df, ncp=0`
Exponential	`exp`	`rate=1`
F Distribution	`f`	`df1, df2`
Gamma	`gamma`	`shape, rate=1 (scale=1/rate)`
Logistic	`logis`	`location=0, scale=1`
Log-normal	`lnorm`	`meanlog=0, sdlog=1`
Normal	`norm`	`mean=0, sd=1`
t Distribution	`t`	`df, ncp=0`
Uniform	`unif`	`min=0, max=1`
Weibull	`weibull`	`shape, scale=1`

Discrete:

Binomial	`binom`	`size, prob`
Geometric	`geom`	`prob`
Hypergeometric	`hyper`	`m, n, k`
Multinomial	`multinom`	`size, prob`
Negative Binomial	`nbinom`	`size, prob, mu`
Poisson	`pois`	`lambda`

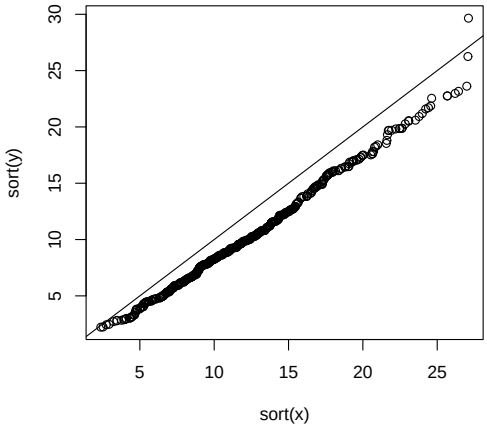
Quantile-Quantile Plot: Same-Size Samples

```
> x <- rgamma(500,shape=6,scale=2)
> y <- rgamma(500,shape=6,scale=2)
> plot(x,y)
> plot(sort(x),sort(y))
> abline(0,1)
>
```



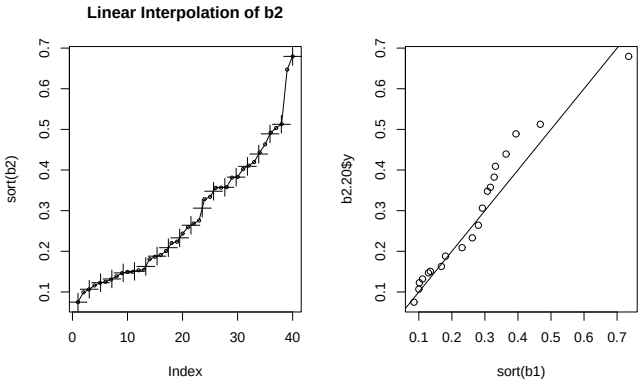
Quantile-Quantile Plot: Same-Size Samples

```
> x <- rgamma(500,shape=6,scale=2)
> y <- rgamma(500,shape=5,scale=2)  # note different shape
> plot(sort(x),sort(y))
> abline(0,1)
>
```



Quantile-Quantile Plot: Different-Size Samples

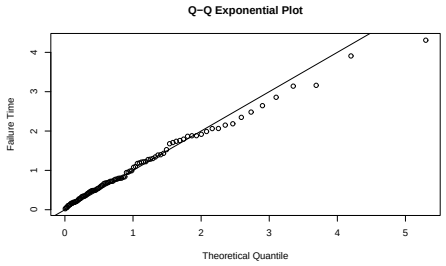
```
> b1 <- rbeta(20,shape1=2,shape2=5)
> b2 <- rbeta(40,shape1=2,shape2=5)
> b2.20 <- approx(sort(b2),n=20)
> plot(sort(b2),type="o",cex=.5,main="Linear Interpolation of b2")
> points(b2.20$x,b2.20$y,pch=3,cex=2)
> plot(sort(b1),b2.20$y)
> abline(0,1)
>
```



Using `qqplot(b1,b2); abline(0,1)` would have produced the same plot.

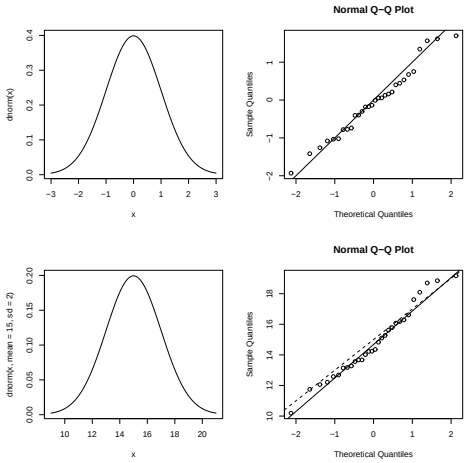
Quantile-Quantile Plot: Compare to Theoretical Dist

```
> ppoints
function (n, a = ifelse(n <= 10, 3/8, 1/2))
{
    if (length(n) > 1)
        n <- length(n)
    if (n > 0)
        (1:n - a)/(n + 1 - 2 * a)
    else numeric(0)
}
> ppoints(20)
 [1] 0.025 0.075 0.125 0.175 0.225 0.275 0.325 0.375 0.425 0.475
[11] 0.525 0.575 0.625 0.675 0.725 0.775 0.825 0.875 0.925 0.975
> times <- rexp(100,rate=.001)
> plot(qexp(ppoints(times)),sort(times/mean(times)),
     main="Q-Q Exponential Plot",
     ylab="Failure Time", xlab="Theoretical Quantile")
> abline(0,1)
>
```



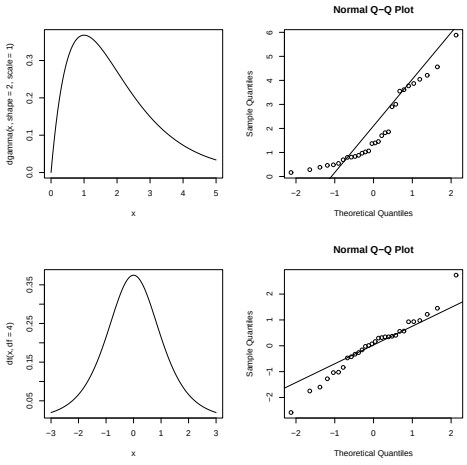
Quantile-Quantile Normal

```
> curve(dnorm(x),-3,3)
> qqnorm(rnorm(30))
> abline(0,1)
> curve(dnorm(x,mean=15,sd=2),15-2*3,15+2*3)
> qqnorm(sample <- rnorm(30,mean=15,sd=2))
> abline(15,2,lty="dashed")
> qqline(sample)
```



Quantile-Quantile Normal

```
> curve(dgamma(x,shape=2,scale=1),0,5)
> qqnorm(sample <- rgamma(30,shape=2,scale=1))
> qqline(sample)
> curve(dt(x,df=4),-3,3)
> qqnorm(sample <- rt(30,df=4))
> qqline(sample)
```



Fitting Univariate Distributions

```
> library(MASS)
> x <- rnorm(100,mean=32,sd=6.2)
> fitdistr(x,"normal")
      mean          sd
  32.8930230    6.7429901
 ( 0.6742990) ( 0.4768014)
> x <- rgamma(100,shape=2.5,rate=6)
> fitdistr(x,"gamma")
     shape       rate
  3.6176870   7.6321941
 (0.4898611) (1.1086225)
> fitdistr(x,"gamma",shape=2.5)
     rate
  5.2743224
 (0.3335774)
Warning message: one-diml optimization by Nelder-Mead is unreliable:
use optimize in: optim(start, mylogfn, x = x, hessian = TRUE, ...)
> fitdistr(x,"gamma",rate=6)
     shape
  2.9447119
 (0.1573992)
Warning message: one-diml optimization by Nelder-Mead is unreliable:
use optimize in: optim(start, mylogfn, x = x, hessian = TRUE, ...)
>
```

(Poorly) Fitting the t Distribution

```
> fitdistr(rt(100,df=5),"t")
       m             s            df
   0.1280011    1.3145470   14.9209134
 ( 0.1399965) ( 0.1815344) (25.9801019)
> fitdistr(rt(100,df=5),"t")
       m             s            df
  0.05308042   0.82689901   4.57266232
 (0.09647543) (0.10220995) (2.12022592)
Warning message:
NaNs produced in: log(x)
> fitdistr(rt(100,df=5),"t")
        m             s             df
  -0.06649982    1.08322211    4.33767086
 ( 0.12674115) ( 0.15053879) ( 2.20698993)
> fitdistr(rt(100,df=5),"t")
        m             s             df
  1.533845e-01   1.387969e+00   1.396458e+05
 (1.387979e-01) (9.815363e-02) (4.843165e+03)
>
```


Generating Random Numbers

- From any supported (univariate) distribution:

```
> rnorm(3)
[1]  2.6527311 -1.7975851 -0.4912465
> rnorm(3)
[1] -1.7124846 -0.5033372 -0.2457211
> rnorm(3,mean=5000,sd=50)
[1] 4934.942 5017.293 5000.024
> rgamma(1,shape=2,scale=6)
[1] 14.27924
> rpois(5,lambda=60)
[1] 58 57 65 66 61
>
```

- Uniform random numbers:
  - Uniform on real interval $[a, b]$,

```
> runif(5, min=-4, max=8)
[1]  4.824822  2.196409 -2.250274 -3.891445  5.604381
```

  - Uniform on $\{0, 1, 2, \ldots, n-1\}$,

```
> floor(runif(15, max=10))
 [1] 0 1 8 4 9 1 9 6 6 6 9 0 0 8 7
```

  - Uniform on $\{1, 2, 3, \ldots, n\}$,

```
> floor(runif(12, max=10))+1
 [1]  3 10  6  8  2  3  8  5 10 10  1  8
```

  - Uniform on $\{m, m+1, \ldots, n\}$,

```
> m <- -2
> n <- 3
> floor(runif(15, min=m, max=n+1))
 [1]  2  0  2 -1  1  2  3 -2  0  1  2 -2  2  0  3
>
```


Using Parameter Vectors

Some examples:

- Every fourth observation has a different parameter (using parameter "recycling"):

```
> rpois(10, lambda=c(1,1,1,100))
 [1]   0   0   1 101   2   2   0  96   1   0
>
```

- Mixture distribution: each observation independently drawn from $N(5, 1)$ with prob 0.2 and from $N(4, 3)$ with prob 0.8.

```
> (coin <- rbinom(20,size=1,prob=.2))
 [1] 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0
> ifelse(coin,5,4)
 [1] 4 5 4 4 4 4 5 4 4 4 5 4 4 4 4 4 5 4 4 4
> rnorm(20, mean=ifelse(coin,5,4), sd=ifelse(coin,1,3))
 [1]  6.7428421  3.3436197  2.8704465  3.6187790
[ . . . ]
[19]  1.6189685  6.4631923
>
```

- Generating a linear regression example:

```
> age <- floor(runif(10, min=5, max=13)) # ages 5-12
> gender <- factor(c("F","M")[rbinom(10,size=1,prob=.5)+1])
> height <- round(rnorm(10,
          mean = 60+6.5*age+5*(gender=="M"), sd=5))
> summary(lm(height ~ gender + age))
[ . . . ]
(Intercept)  68.6676     6.7869  10.118 1.98e-05 ***
genderM       1.6892     3.1540   0.536    0.609
age           5.7162     0.6883   8.305 7.17e-05 ***
[ . . . ]
Residual standard error: 4.586 on 7 degrees of freedom
[ . . . ]
```


Random Sampling

- Random permutation (i.e., perfect shuffle)

```
> sample(10)   # sample(1:10) has same effect
 [1]  6  9  2  7  5 10  4  3  1  8
> sample(c("Stan","Ruojia","Sam","Ahmed","Felicia"))
[1] "Stan"    "Sam"     "Ruojia"  "Felicia" "Ahmed"
> iris[sample(nrow(iris)),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
43           4.4         3.2          1.3         0.2     setosa
115          5.8         2.8          5.1         2.4  virginica
[ . . . 147 more randomly ordered records . . . ]
92           6.1         3.0          4.6         1.4 versicolor
```

- Random subsample (without replacement)

```
> dim(iris)
[1] 150   5
> iris.sample <- iris[sample(nrow(iris),size=30),]
> dim(iris.sample)
[1] 30  5
```

- Random subsample (with replacement)

```
> sample(10,size=5)               # without replacement
[1] 8 6 3 2 7
> sample(10,size=5,replace=T)     # with replacement
[1] 3 4 9 6 6
> sample(10, size=20)
Error in sample(x, size, replace, prob) : can't take a
sample larger than the population when replace = FALSE
> sample(10, size=20, replace=T)
 [1]  8  6  9  5  4  9  1  6  3  6  7  5  4  9  9  8  8  2  8 10
>
```

The Random Number Seed

```
> rnorm(5)
[1] -0.53931674  0.51708133  0.02273572  0.68687194 -0.16931581
> rnorm(5)
[1] -0.4644731 -1.4228934  0.1793277 -0.1587798  0.2894271
> set.seed(621)
> rnorm(5)
[1]  0.6832647 -0.8720841  1.4884790 -0.3136084 -1.0155666
> rnorm(5)
[1] -0.4917928 -2.4038981  0.2535542  0.4300435 -0.3850864
> set.seed(621)
> rnorm(5)
[1]  0.6832647 -0.8720841  1.4884790 -0.3136084 -1.0155666
> rnorm(5)
[1] -0.4917928 -2.4038981  0.2535542  0.4300435 -0.3850864
> set.seed(621)
> rnorm(10)
 [1]  0.6832647 -0.8720841  1.4884790 -0.3136084 -1.0155666
 [6] -0.4917928 -2.4038981  0.2535542  0.4300435 -0.3850864
>

> RNGkind()
[1] "Mersenne-Twister" "Inversion"
> .Random.seed
  [1]         403          20  -934327767  1112312283
[ . . . ]
[625] -1627563568  1404439705
>
```


Data Summaries: Quantiles

```
> library(MASS)
> abbey
 [1]   5.2   6.5   6.9   7.0   7.0   7.0
 [7]   7.4   8.0   8.0   8.0   8.0   8.5
[13]   9.0   9.0  10.0  11.0  11.0  12.0
[19]  12.0  13.7  14.0  14.0  14.0  16.0
[25]  17.0  17.0  18.0  24.0  28.0  34.0
[26] 125.0
> summary(abbey)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.20    8.00   11.00   16.01   15.00  125.00
> min(abbey)
[1] 5.2
> max(abbey)
[1] 125
> range(abbey)
[1]   5.2 125.0
> diff(range(abbey))
[1] 119.8
> median(abbey)
[1] 11
> quantile(abbey, .30)
30%
  8
> quantile(abbey, c(0,.05,.25,.5,.75,.95,1))
   0%    5%   25%   50%   75%   95%  100%
  5.2   6.7   8.0  11.0  15.0  31.0 125.0
> quantile(abbey)
   0%   25%   50%   75%  100%
  5.2   8.0  11.0  15.0 125.0
> IQR(abbey)
[1] 7
>
```


Data Summaries: Univariate Stats

Sample mean	$\bar{x} = \sum_{i=1}^{n} x_i$	`> mean(abbey)` `[1] 16.00645`
Sample variance	$s_X^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$	`> var(abbey)` `[1] 452.3733`
Sample std. dev.	$s_X = \sqrt{s_X^2}$	`> sd(abbey)` `[1] 21.26907`
Sample covariance	$s_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$	`> cov(bp, pres)` `[1] 17.34638`
Sample correlation	$r_{XY} = \frac{s_{XY}}{\sqrt{s_X^2 s_Y^2}}$	`> cor(bp, pres)` `[1] 0.9972102`

Boxplots!!

- Graphical display of Tukey's five number summary: minimum, lower hinge F_L, median, upper hinge F_U, and maximum.

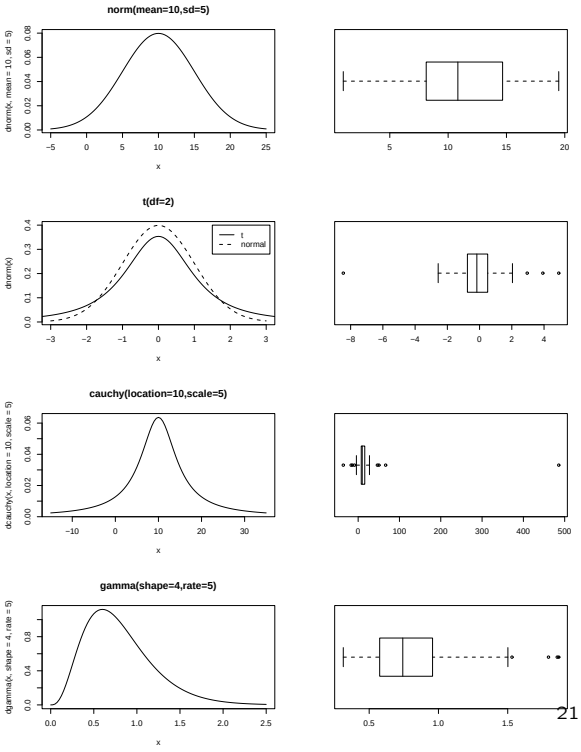
```
> quantile(abbey)
   0%   25%   50%   75%  100%
  5.2   8.0  11.0  15.0 125.0
> fivenum(abbey)
[1]   5.2   8.0  11.0  15.0 125.0
> sample <- 1:4
> quantile(sample)
  0%  25%  50%  75% 100%
1.00 1.75 2.50 3.25 4.00
> fivenum(sample)
[1] 1.0 1.5 2.5 3.5 4.0
>
```

- However, anything outside the range

$$F_L - 1.5\,d_F \quad \text{and} \quad F_U + 1.5\,d_F$$

(where $d_F = F_U - F_L$ is the interhinge range) is considered an outlier and plotted separately.

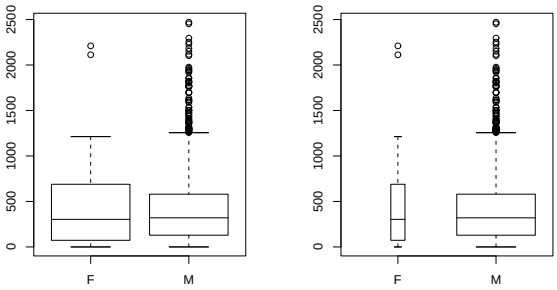
Samples ($n = 50$) from Some Dists



Interesting Variations

- Box widths proportional to $\sqrt{n}$:

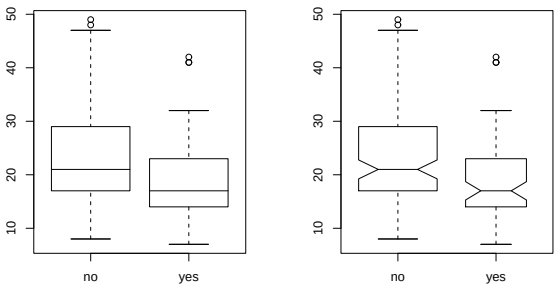
```
> library(MASS)
> attach(Aids2)
> names(Aids2)
[1] "state"   "sex"     "diag"    "death"   "status"
[6] "T.categ" "age"
> boxplot((death-diag) ~ sex)
> boxplot((death-diag) ~ sex, varwidth=T)
> table(sex)
sex
   F    M
  89 2754
>
```



Interesting Variations

- Box notches: no overlap ⇒ very likely difference

```
> library(MASS)
> boxplot(y ~ limit, data=Traffic)
> boxplot(y ~ limit, data=Traffic, varwidth=T, notch=T)
>
```

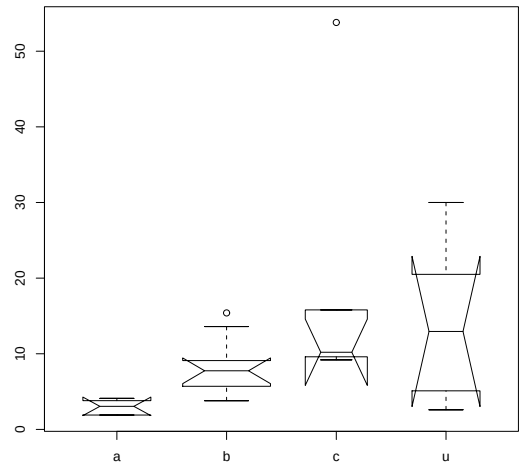


Notches drawn to span $median \pm 1.58\, IHR/\sqrt{n}$ (a little bigger than an approximate 90% CI for the median).

Interesting Variations

- Another box-notch and variable width example:

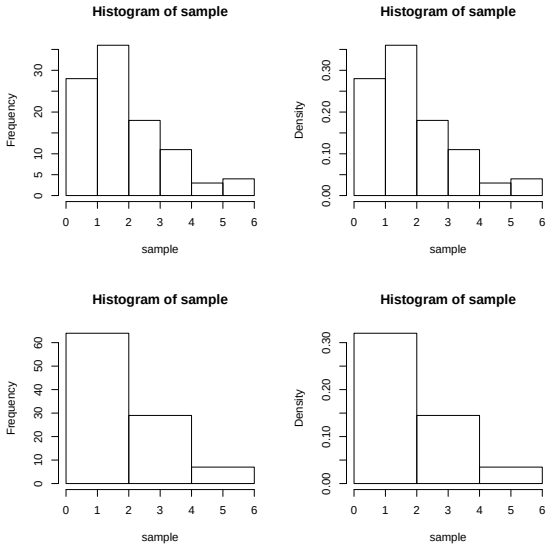
```
> library(MASS)
> boxplot(Tetrahydrocortisone ~ Type, data=Cushings,
          varwidth=T, notch=T)
>
```



Histograms!

. . . by count (`freq=T`) or density (`freq=F`)

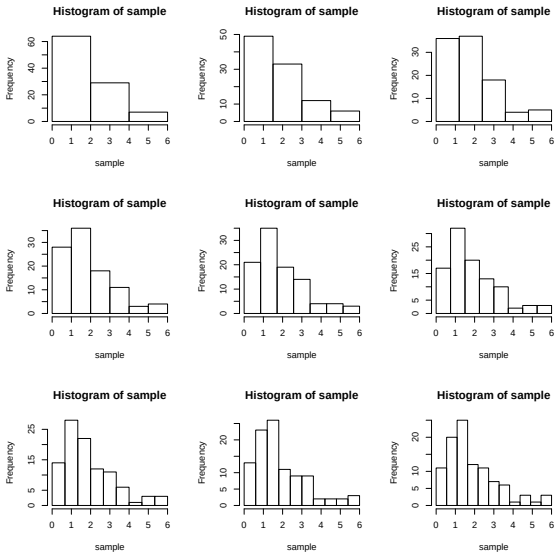
```
> sample <- rgamma(100, shape=2, rate=1)
> hist(sample)          # box height is count
> hist(sample, freq=F)  # box area is proportion of obs
> hist(sample, breaks=3)
> hist(sample, freq=F, breaks=3)
```



Histograms!

. . . are sensitive to choice of breaks.

```
> opar <- par(mfrow=c(3,3))
> for (i in 4:12) hist(sample,breaks=seq(from=0,to=6,length=i))
> par(opar)
```



Histogram Breaks

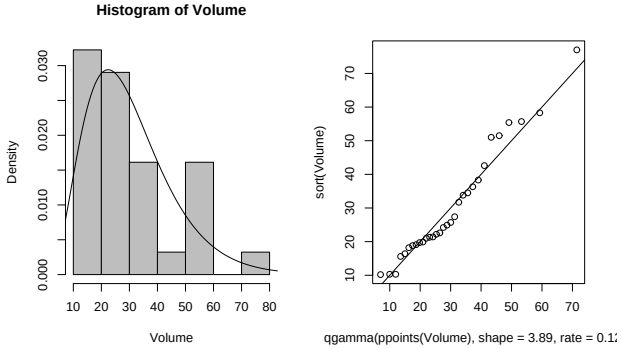
The parameter `breaks` can be given as

- a *suggested* number of breaks `breaks=10`;
- the name of an algorithm to generate a *suggested* number of breaks:
  - `"sturges"`: number of breaks is $\log_2(n) + 1$
  - `"scott"`: box width is $3.5sn^{-1/3}$
  - `"fd"`: box width is $2\,IQR\,n^{-1/3}$

  This default is `"sturges"` which looks okay for "normalish" data but isn't resistant to outliers.
- a vector of exact breaks (which may be unequally spaced).

Histograms w/ Density Curves

```
> library(MASS)  # needed for fitdistr
> attach(trees)
> hist(Volume, freq=F, col="grey")
> fitdistr(Volume,"gamma")
     shape         rate
  3.88581759   0.12880083
[ . . . ]
> curve(dgamma(x, shape=3.89, rate=0.129),add=T)
> plot(qgamma(ppoints(Volume), shape=3.89, rate=0.129), sort(Volume))
> abline(0,1)
>
```



Stem and Leaf Plot

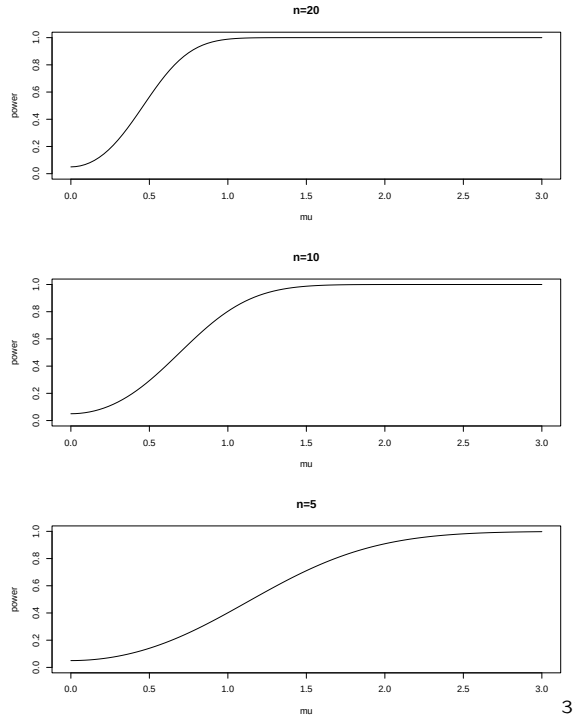
```
> library(MASS)
> abbey
 [1]   5.2   6.5   6.9   7.0   7.0   7.0
 [7]   7.4   8.0   8.0   8.0   8.0   8.5
[13]   9.0   9.0  10.0  11.0  11.0  12.0
[19]  12.0  13.7  14.0  14.0  14.0  16.0
[25]  17.0  17.0  18.0  24.0  28.0  34.0
[31] 125.0
> round(abbey)   # see help(round) about funny IEEE rounding
 [1]   5   6   7   [ . . . ]
> floor(abbey+.5)  # or round(abbey+.0000001)
 [1]   5   7   7   [ . . . ]
> stem(abbey)

  The decimal point is 1 digit(s) to the right of the |

   0 | 577777788889990112244446778
   2 | 484
   4 |
   6 |
   8 |
  10 |
  12 | 5

> stem(abbey[abbey != 125])

  The decimal point is 1 digit(s) to the right of the |

  0 | 57777778888999
  1 | 011224444
  1 | 6778
  2 | 4
  2 | 8
  3 | 4

>
```


One-Sample t-Test

```
> t.test(rnorm(20,mean=2,sd=2))

One Sample t-test

data:  rnorm(20, mean = 2, sd = 2)
t = 2.384, df = 19, p-value = 0.02771
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.1558692 2.3985594
sample estimates:
mean of x
 1.277214

> t.test(rnorm(20,mean=0,sd=2))

One Sample t-test

data:  rnorm(20, mean = 0, sd = 2)
t = -0.9111, df = 19, p-value = 0.3737
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -1.5884003  0.6249483
sample estimates:
mean of x
-0.481726

>
```

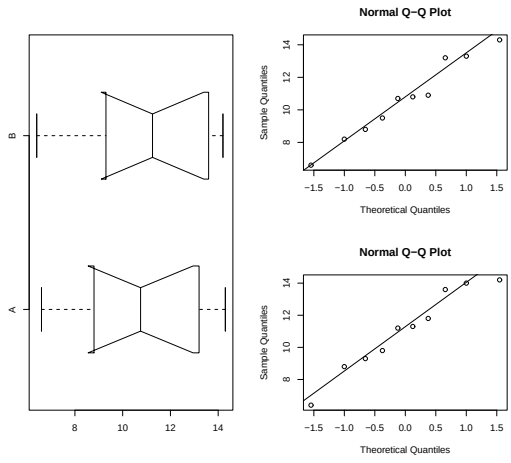

Power Curves for t-test

$\alpha = .05$, $X_1, \ldots, X_n \sim N(\mu, 1)$



Shoe Wear

```
> library(MASS)
> shoes
$A
 [1] 13.2  8.2 10.9 14.3 10.7  6.6  9.5 10.8  8.8 13.3
$B
 [1] 14.0  8.8 11.2 14.2 11.8  6.4  9.8 11.3  9.3 13.6
> layout(rbind(c(1,2),c(1,3)))
> boxplot(list(A=shoes$A, B=shoes$B),
          notch=T, horizontal=T, boxwex=.5)
> qqnorm(shoes$A); qqline(shoes$A)
> qqnorm(shoes$B); qqline(shoes$B)
```



Paired t-Test

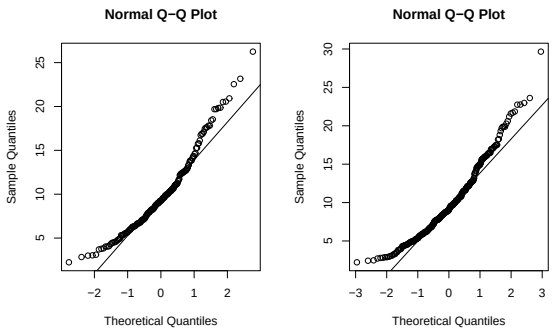
```
> t.test(shoes$A, shoes$B, paired=T)

Paired t-test

data:  shoes$A and shoes$B
t = -3.3489, df = 9, p-value = 0.008539
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.6869539 -0.1330461
sample estimates:
mean of the differences
                 -0.41

> t.test(shoes$A-shoes$B)

One Sample t-test

data:  shoes$A - shoes$B
t = -3.3489, df = 9, p-value = 0.008539
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.6869539 -0.1330461
sample estimates:
mean of x
    -0.41

>
```


Paired vs. Unpaired

```
> t.test(shoes$A, shoes$B, paired=T)

Paired t-test

data:  shoes$A and shoes$B
t = -3.3489, df = 9, p-value = 0.008539
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.6869539 -0.1330461
sample estimates:
mean of the differences
                 -0.41

> t.test(shoes$A, shoes$B)  # bad idea: using paired=F

Welch Two Sample t-test

data:  shoes$A and shoes$B
t = -0.3689, df = 17.987, p-value = 0.7165
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.745046  1.925046
sample estimates:
mean of x mean of y
    10.63     11.04

>
```

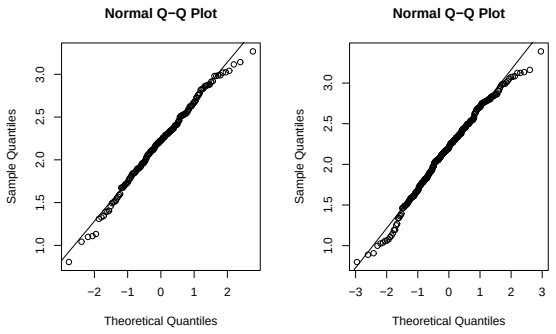

Non-normality of Traffic Data

```
> library(MASS)
> attach(Traffic)
> qqnorm(y[limit=="yes"]); qqline(y[limit=="yes"])
> qqnorm(y[limit=="no"]); qqline(y[limit=="no"])
> shapiro.test(y[limit=="yes"])
Shapiro-Wilk normality test
data:  y[limit == "yes"]
W = 0.9213, p-value = 0.0003330
> shapiro.test(y[limit=="no"])
Shapiro-Wilk normality test
data:  y[limit == "no"]
W = 0.9516, p-value = 0.0003928
>
```



Normality (?) of Log Traffic Data

```
> qqnorm(log(y[limit=="yes"]))
> qqnorm(log(y[limit=="no"]))
> shapiro.test(log(y[limit=="yes"]))
Shapiro-Wilk normality test
data:  log(y[limit == "yes"])
W = 0.9832, p-value = 0.4814
> shapiro.test(log(y[limit=="no"]))
Shapiro-Wilk normality test
data:  log(y[limit == "no"])
W = 0.9868, p-value = 0.3235
>
```



Two-Sample t-Test

```
> ly.yes <- log(y[limit=="yes"])
> ly.no <- log(y[limit=="no"])
> t.test(ly.yes, ly.no)
Welch Two Sample t-test
data:  ly.yes and ly.no
t = -3.2954, df = 147.673, p-value = 0.001231
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.31472006 -0.07876154
sample estimates:
mean of x mean of y
 2.866770  3.063511
> var.test(ly.yes, ly.no)
F test to compare two variances
data:  ly.yes and ly.no
F = 0.9262, num df = 68, denom df = 114, p-value = 0.7389
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6109935 1.4372433
sample estimates:
ratio of variances
         0.9262063
> t.test(ly.yes, ly.no, var.equal=T)
Two Sample t-test
data:  ly.yes and ly.no
t = -3.2638, df = 182, p-value = 0.001313
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.31567636 -0.07780524
sample estimates:
mean of x mean of y
 2.866770  3.063511
>
```


Approximate Z-Tests in R

Instead of log-transforming traffic data, let's do an approximate Z-test on the untransformed data:

- Same statistic, so use same test function:

```
> t.test(y[limit=="yes"],y[limit=="no"])

 Welch Two Sample t-test

data:  y[limit == "yes"] and y[limit == "no"]
t = -3.3995, df = 165.545, p-value = 0.000846
alternative hypothesis: true difference
in means is not equal to 0
95 percent confidence interval:
 -6.666816 -1.767967
sample estimates:
mean of x mean of y
 18.91304  23.13043
```

- If you want to be picky, use p-value based on Z instead of $t(165.545)$:

```
> 2*pt(-3.3995,df=165.545,lower.tail=T)
[1] 0.0008459695
> 2*pnorm(-3.3995,lower.tail=T)
[1] 0.0006750918
>
```

  but this is all so approximate anyway, why not take the more conservative t-based p-value?

Recall that "exact" p-value for log-transformed data assuming normality and equal variances was $p = 0.001313$.

Wilcoxon Signed-Rank Test

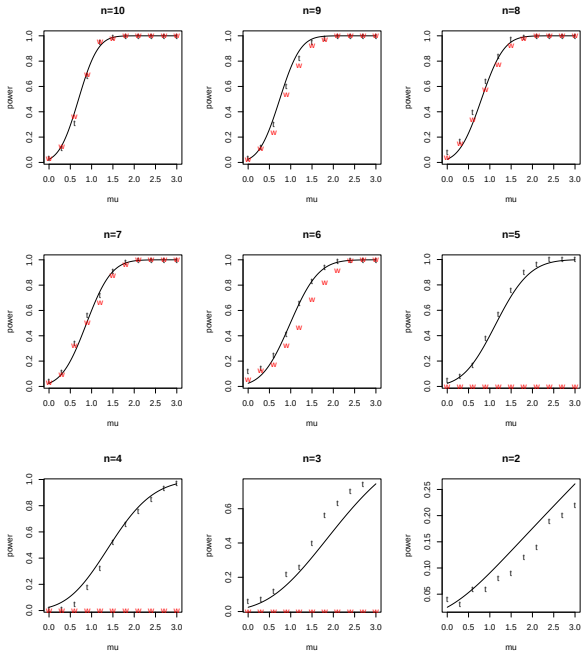
- For one sample:

```
> x <- c(8.5,8.6,6.4,12.1,8.2,7.4,7.8,8.3,10.3,8.4)
> wilcox.test(x, mu=10, conf.int=T)
Wilcoxon signed rank test
data:  x
V = 8, p-value = 0.04883
alternative hypothesis: true mu is not equal to 10
95 percent confidence interval:
 7.50 9.95
sample estimates:
(pseudo)median
          8.35
```

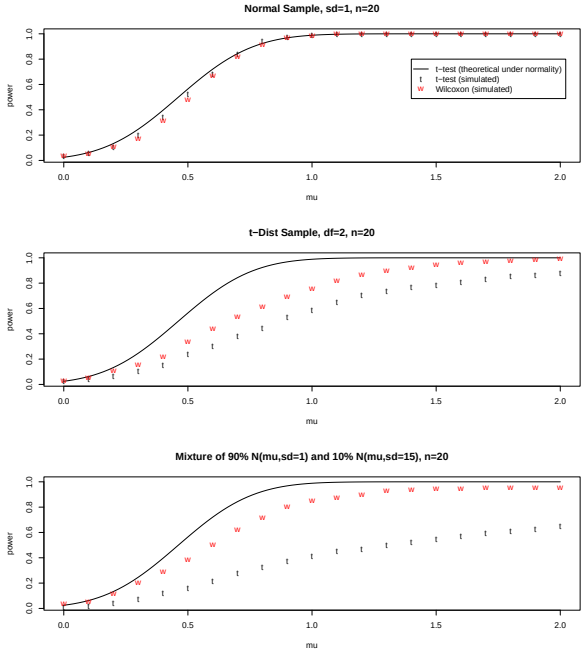
- For paired data:

```
> wilcox.test(shoes$A, shoes$B, paired=T)
Wilcoxon signed rank test with continuity correction
data:  shoes$A and shoes$B
V = 3, p-value = 0.01431
alternative hypothesis: true mu is not equal to 0
Warning message: Cannot compute exact p-value with ties in:
wilcox.test.default(shoes$A, shoes$B, paired = T)
> wilcox.test(shoes$A-shoes$B)
Wilcoxon signed rank test with continuity correction
data:  shoes$A - shoes$B
V = 3, p-value = 0.01431
alternative hypothesis: true mu is not equal to 0
Warning message: Cannot compute exact p-value with ties in:
wilcox.test.default(shoes$A - shoes$B)
>
```


Why I Love the Wilcoxon Test



Why I Love It Even More



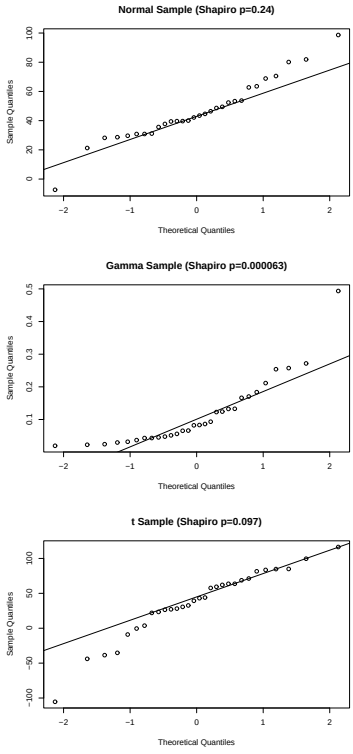
Two-Sample Wilcoxon Test

```
> wilcox.test(y.yes,y.no)

Wilcoxon rank sum test with continuity correction

data:  y.yes and y.no
W = 2878, p-value = 0.001830
alternative hypothesis: true mu is not equal to 0

> wilcox.test(log(y.yes),log(y.no))

Wilcoxon rank sum test with continuity correction

data:  log(y.yes) and log(y.no)
W = 2878, p-value = 0.001830
alternative hypothesis: true mu is not equal to 0

>
```


Power of Shapiro Test

```
> rejects <- 0
> for (i in 1:10000) {
+   x <- rnorm(30, mean=45, sd=20)
+   if (shapiro.test(x)$p.value < 0.05) rejects <- rejects + 1
+ }
> rejects
[1] 530
> rejects <- 0
> for (i in 1:10000) {
+   x <- rgamma(30, shape=2, rate=20)
+   if (shapiro.test(x)$p.value < 0.05) rejects <- rejects + 1
+ }
> rejects
[1] 7513
> rejects <- 0
> for (i in 1:10000) {
+   x <- 30+45*rt(30, df=4)
+   if (shapiro.test(x)$p.value < 0.05) rejects <- rejects + 1
+ }
> rejects
[1] 3249
>
```


Power of Shapiro Test



Multigroup Location Tests

```
> names(PlantGrowth)
[1] "weight" "group"
> levels(PlantGrowth$group)
[1] "ctrl" "trt1" "trt2"
> oneway.test(weight ~ group, data=PlantGrowth)

One-way analysis of means (not assuming equal variances)

data:  weight and group
F = 5.181, num df = 2.000, denom df = 17.128, p-value = 0.01739

> oneway.test(weight ~ group, data=PlantGrowth, var.equal=T)

One-way analysis of means

data:  weight and group
F = 4.8461, num df = 2, denom df = 27, p-value = 0.01591

> summary(aov(weight ~ group, data=PlantGrowth))
            Df  Sum Sq Mean Sq F value  Pr(>F)
group        2  3.7663  1.8832  4.8461 0.01591 *
Residuals   27 10.4921  0.3886
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> kruskal.test(weight ~ group, data=PlantGrowth)

Kruskal-Wallis rank sum test

data:  weight by group
Kruskal-Wallis chi-squared = 7.9882, df = 2, p-value = 0.01842

>
```


Testing Two-Way Contingencies

```
> table(r,stat404)
   stat404
r   -  n  y
  0  2  4  0
  1  3  4  1
  2  0  7 14
  3  1  1 10
> chisq.test(table(r,stat404))

Pearson’s Chi-squared test

data:  table(r, stat404)
X-squared = 21.9431, df = 6, p-value = 0.00124

Warning message:
Chi-squared approximation may be incorrect
in: chisq.test(table(r, stat404))
> fisher.test(table(r,stat404))

Fisher’s Exact Test for Count Data

data:  table(r, stat404)
p-value = 0.0001139
alternative hypothesis: two.sided
>
```


Testing Two-Way Contingencies

```
> table(matlab,stat404)
      stat404
matlab -  n  y
     0  5  9 17
     1  0  6  5
     2  1  1  1
     3  0  0  2
> chisq.test(table(matlab,stat404))

Pearson’s Chi-squared test

data:  table(matlab, stat404)
X-squared = 6.3824, df = 6, p-value = 0.3817

Warning message:
Chi-squared approximation may be incorrect
in: chisq.test(table(matlab, stat404))
> fisher.test(table(matlab,stat404))

Fisher’s Exact Test for Count Data

data:  table(matlab, stat404)
p-value = 0.3786
alternative hypothesis: two.sided

>
```


Bigger Example

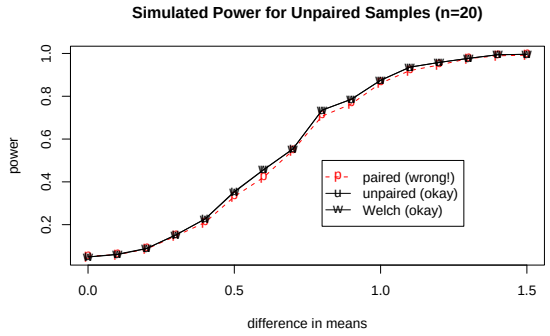
```
> library(MASS)
> names(Melanoma)
[1] "time"      "status"    "sex"       "age"       "year"
[6] "thickness" "ulcer"
> attach(Melanoma)
> table(sex,ulcer)
   ulcer
sex 0  1
  0 79 47
  1 36 43
> chisq.test(table(sex,ulcer))

Pearson’s Chi-squared test with Yates’ continuity correction

data:  table(sex, ulcer)
X-squared = 5.1099, df = 1, p-value = 0.02379

> fisher.test(table(sex,ulcer))

Fisher’s Exact Test for Count Data

data:  table(sex, ulcer)
p-value = 0.02061
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 1.089871 3.700065
sample estimates:
odds ratio
  2.000701

>
```

For Unpaired (Independent) Samples

Independent, iid samples:

$$X_1, \ldots, X_{20} \sim N(\mu_X, 1)$$
$$Y_1, \ldots, Y_{20} \sim N(\mu_Y, 1)$$

At $\alpha = 0.05$,

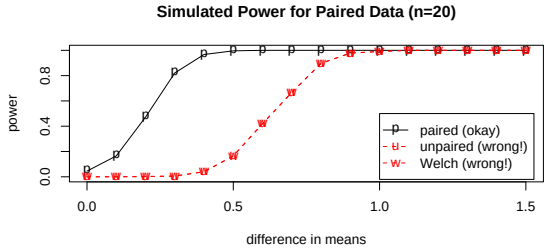


For Paired Data

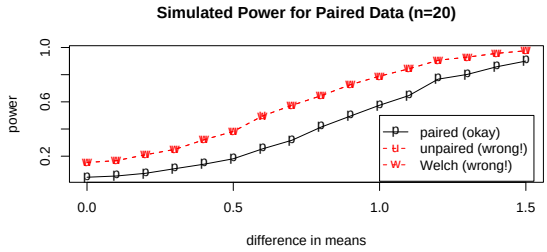
$(X_1, Y_1), \ldots, (X_{20}, Y_{20})$ iid bivariate normal.

At $\alpha = 0.05$,

- Positive correlation (usual) $\rho = +0.9$



- Negative correlation (unusual) $\rho = -0.9$



Summary of Tests

Location tests:

- `t.test`: t-tests for one sample, paired, two-sample (equal variance and Welch's approximation). Can also be used for approximate Z-tests.
- `wilcox.test`: Wilcoxon signed-rank test for one sample or paired data, Wilcoxon/Mann-Whitney rank-sum test for two samples.
- `oneway.test`: F-test for difference in means of two or more samples (equal variance and Welch's approximation).
- `kruskal.test`: Kruskal-Wallis rank-sum test for difference in means of two or more samples.

Categorical data tests:

- `chisq.test`: χ^2-test for one-way and interaction in two-way contingency tables.
- `fisher.test`: Fisher's exact test for interaction in two-way contingency tables.

Other tests:

- `shapiro.test`: Shapiro-Wilk test for H_0 : sample is normal.
- `var.test`: F-test for H_0 : samples have equal variances.

Relative Efficiency of $\tilde{X}$ and $\bar{X}$

```
> means <- medians <- numeric(0)  # make empty vectors
> for (i in 1:1000) {
+   x <- rnorm(30,mean=10,sd=5)
+   means[i] <- mean(x)
+   medians[i] <- median(x)
+ }
> var(means)
[1] 0.8111457
> var(medians)
[1] 1.228001
> var(means)/var(medians)  # simulated RE, normal case
[1] 0.6605414
> for (i in 1:1000) {
+   x <- 10+5*rt(30,df=5)
+   means[i] <- mean(x)
+   medians[i] <- median(x)
+ }
> var(means)
[1] 1.350055
> var(medians)
[1] 1.319712
> var(means)/var(medians)  # simulated RE, t(5) case
[1] 1.022992
>
```

Trimmed Mean

```
> x <- 10+5*rt(70,df=3)  # sample size 70
> mean(x)                # default trim = 0
[1] 11.09950
> mean(x,trim=0)
[1] 11.09950
> mean(x,trim=0.05)      # 5% off each end
[1] 10.34636
> 70*0.05
[1] 3.5                  # that is, 3 obs off each end
> mean(sort(x)[4:67])
[1] 10.34636
> mean(x,trim=0.5)       # trimming off everything...
[1] 10.18221
> median(x)              # ...gives the median
[1] 10.18221
>
```


Efficiency of Trimmed Mean

```
> trims <- seq(0,0.5,length=10)
> trimmeans <- matrix(NA,nrow=1000,ncol=10)
> for (i in 1:1000) {
+   x <- rnorm(30,mean=10,sd=5)
+   for (j in 1:10) {
+     trimmeans[i,j] <- mean(x,trim=trims[j])
+   }
+ }
> trims
 [1] 0.0000000 0.0555556 0.1111111 0.1666667 0.2222222
 [6] 0.2777778 0.3333333 0.3888889 0.4444444 0.5000000
> (vars <- apply(trimmeans,2,var))
 [1] 0.8272082 0.8308127 0.8708342 0.9230062 0.9517421
 [6] 1.0231556 1.0956334 1.1251648 1.1916106 1.2344185
> vars[1]/vars
 [1] 1.0000000 0.9956615 0.9499032 0.8962109 0.8691516
 [6] 0.8084872 0.7550045 0.7351885 0.6941934 0.6701197

[ repeat with x <- 10+5*rt(30,df=5) ]
> vars[1]/vars
 [1] 1.0000000 1.1433810 1.2279008 1.2284159 1.2215654
 [6] 1.1857230 1.1303785 1.0970575 1.0152993 0.9801351

[ repeat with x <- rnorm(30,mean=10,
          sd=ifelse(rbinom(30,1,.05),10,5)) ]
> vars[1]/vars
 [1] 1.0000000 1.0318539 1.0208543 0.9858267 0.9587991
 [6] 0.8952624 0.8441214 0.8151472 0.7547057 0.7171817
>
```


Robust Scale Estimates

- Some scale estimates:

```
> attach(Traffic)
> y.yes <- y[limit=="yes"]; y.no <- y[limit=="no"]
> c(sd(y.yes),sd(y.no))
[1] 7.474937 9.157991
> c(mad(y.yes),mad(y.no))
[1] 7.4130 8.8956
> c(IQR(y.yes)/1.349,IQR(y.no)/1.349)
[1] 6.671609 8.895478
>
```

- Simulated relative efficiency (under normality):

```
> mads <- numeric(0)
> iqrs <- numeric(0)
> sds  <- numeric(0)
> for (i in 1:1000) {
+   x <- rnorm(30,mean=10,sd=5)
+   mads[i] <- mad(x)
+   iqrs[i] <- IQR(x)/1.349
+   sds[i]  <- sd(x)
+ }
> var(sds)/var(mads)
[1] 0.3784513
> var(sds)/var(iqrs)
[1] 0.3923162
>
```


Huber's Estimate

```
> x <- 10+5*rt(70,df=3)
> huber(x,k=10000)
$mu
[1] 11.09950
$s
[1] 6.260165
> mean(x)
[1] 11.09950
> huber(x,k=0)
$mu
[1] 10.18221
$s
[1] 6.260165
> median(x)
[1] 10.18221
> huber(x)           # default k=1.5
$mu
[1] 10.38661
$s
[1] 6.260165
>
```

Huber's Proposal 2

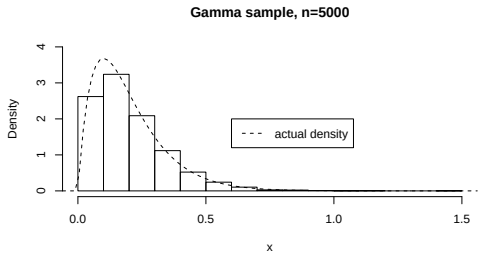
```
> x <- rnorm(20,mean=10,sd=5)
> hubers(x,k=10000)
$mu
[1] 8.769674
$s
[1] 4.816868
> mean(x)
[1] 8.769674
> sd(x)
[1] 4.816868
> hubers(x,k=0.00001)
$mu
[1] 9.667696
$s
[1] 21120.81
> median(x)
[1] 9.667696
> hubers(x)          # default k=1.5
$mu
[1] 8.773504
$s
[1] 5.451605
>
```


Efficiency of Proposal 2

```
> means <- numeric(0)
> sds <- numeric(0)
> hubers.means <- numeric(0)
> hubers.sds <- numeric(0)
> for (i in 1:1000) {
+   x <- rnorm(25,mean=30,sd=10)
+   means[i] <- mean(x)
+   sds[i] <- sd(x)
+   huber1 <- hubers(x)
+   hubers.means[i] <- huber1$mu
+   hubers.sds[i] <- huber1$s
+ }
> var(means)/var(hubers.means)
[1] 0.969263
> var(sds)/var(hubers.sds)
[1] 0.7299158

[ repeat with x <- 30+10*rt(25,df=5) ]
> var(means)/var(hubers.means)
[1] 1.213951
> var(sds)/var(hubers.sds)
[1] 2.442511
>
```


Histogram as Density Estimate

```
> x <- rgamma(5000,shape=2,rate=10)
> hist(x,freq=F,breaks=20,main="Gamma sample, n=5000")
> curve(dgamma(x,shape=2,rate=10),lty=2,add=T)
> legend(0.6,2,c("actual density"),lty=2)
```

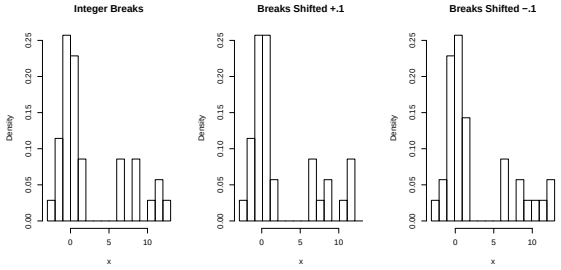


Pros:

- area under curve is 1;
- area between $x = a$ and $x = b$ is, roughly, proportion of observations in $[a, b)$ and so an estimate of $\Pr(a \leq X \leq b)$;
- looks like a pretty good estimate for *big* samples.

Histogram as Density Estimate

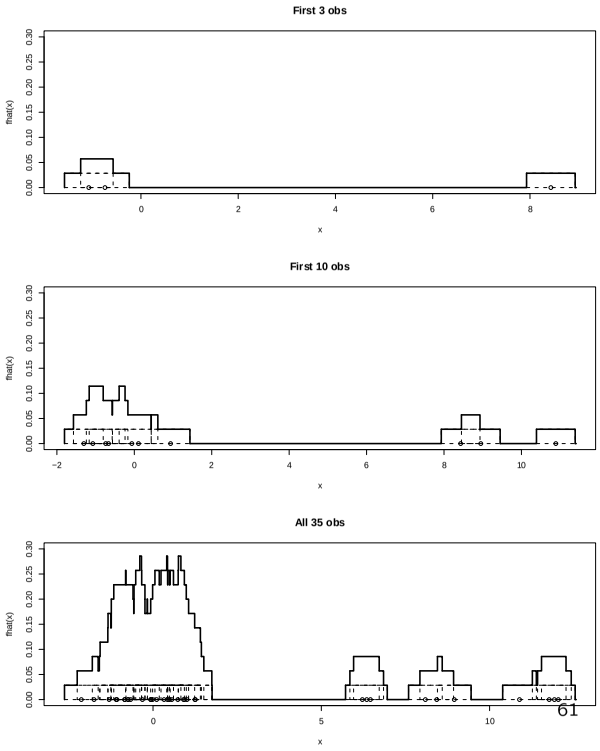
```
> x <- sample(c(rnorm(25,0,1),rnorm(10,10,2)))
> range(x)
[1] -2.129958 12.042381
> breaks <- seq(-3,13,by=1)
> opar <- par(mfrow=c(1,3))
> hist(x,freq=F,breaks=breaks, main="Integer Breaks")
> hist(x,freq=F,breaks=breaks+.1, main="Breaks Shifted +.1")
> hist(x,freq=F,breaks=breaks-.1, main="Breaks Shifted -.1")
> par(opar)
```



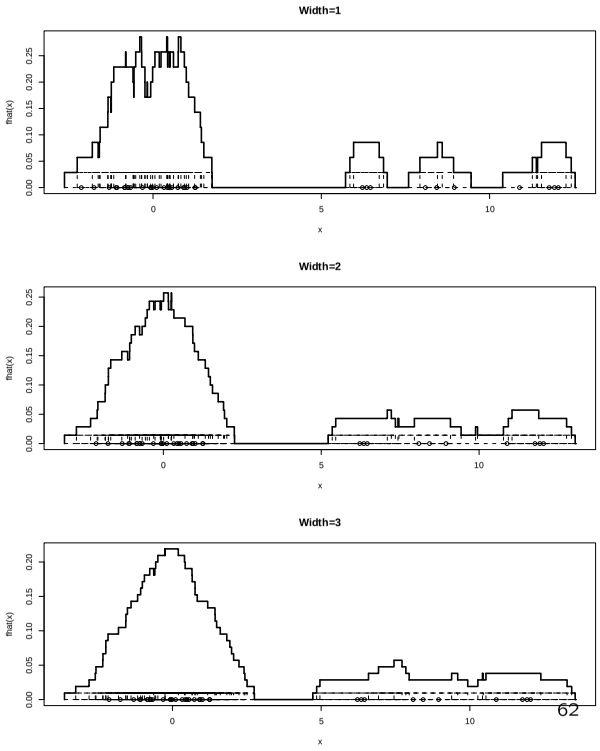
Cons:

- discontinuous and just plain ugly;
- very sensitive to choice of breaks;
- generally poor estimate for moderate sample sizes.

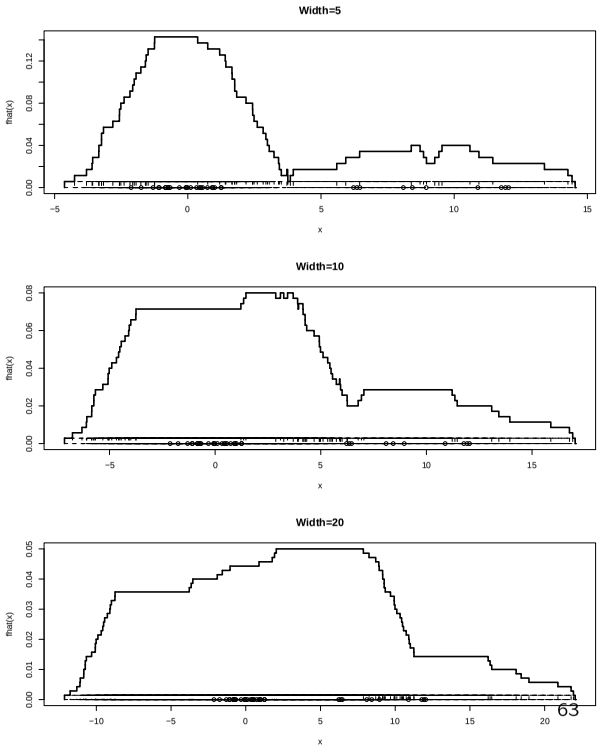
Adding up Little Rectangles



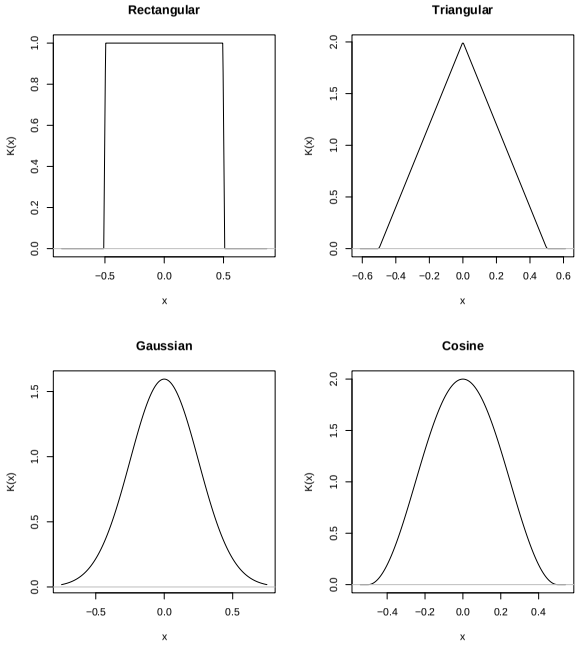
Using Different-Width Rectangles



Using Different-Width Rectangles



Some Different Kernels



The R `density` Function

```
density(x, bw="nrd0", adjust=1, kernel="gaussian")
```

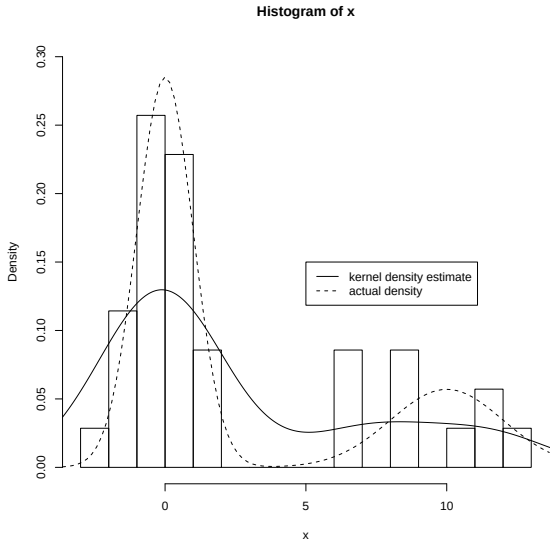
- `kernel` can be `rectangular`, `triangular`, `gaussian`, or `cosine` (or even `epanechnikov`, `biweight`, or `optcosine`);
- `bw` can be (a) a number, and the kernel will be scaled to have this standard deviation; or (b) the name of a special bandwidth-choosing algorithm: `nrd0`, `nrd`, `ucv`, `bcv`, `sj`, or `sj-dpi`;
- `adjust` is a constant to scale a bandwidth chosen by an algorithm.

Note the function returns a `density` object with the most useless default output possible:

```
> density(x)

Call:
     density(x = x)

Data: x (35 obs.); Bandwidth 'bw' = 1.984

       x                y
 Min.   :-8.082   Min.   :0.0001709
 1st Qu.:-1.563   1st Qu.:0.0094211
 Median : 4.956   Median :0.0292758
 Mean   : 4.956   Mean   :0.0383052
 3rd Qu.:11.475   3rd Qu.:0.0456803
 Max.   :17.994   Max.   :0.1297325
>
```

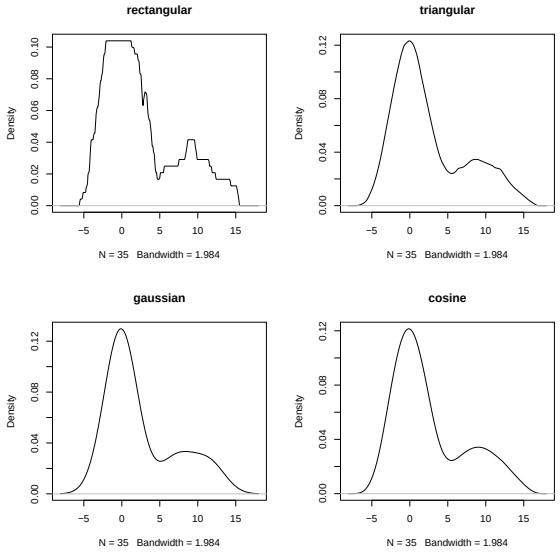

Superimposed on a Histogram

```
> hist(x, freq=F, nclass=20, ylim=c(0,.3))
> lines(density(x))
> curve(25/35*dnorm(x,0,1)+10/35*dnorm(x,10,2), lty=2, add=T)
> legend(5,.15,c("kernel density estimate","actual density"),
+ lty=c(1,2))
```



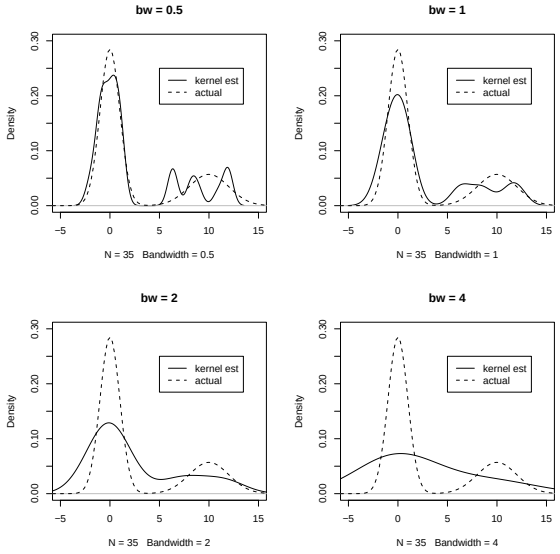
Effect of Kernel

All plots with bandwidth `bw` = 1.984 as chosen by `nrd0`.



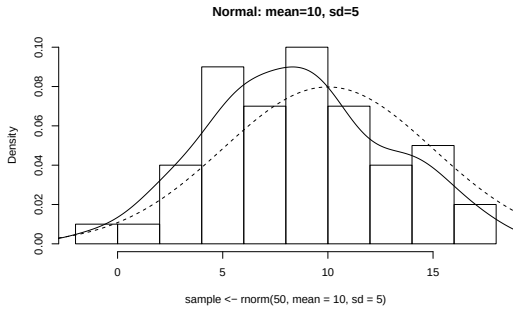
Effect of Bandwidth

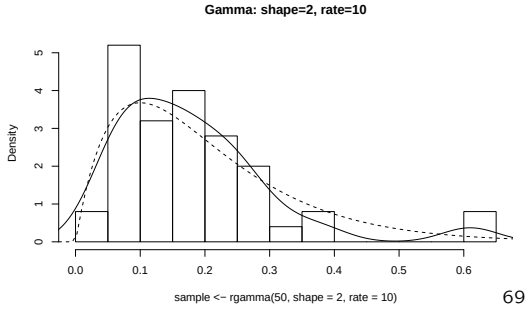
All plots with default Gaussian kernel.



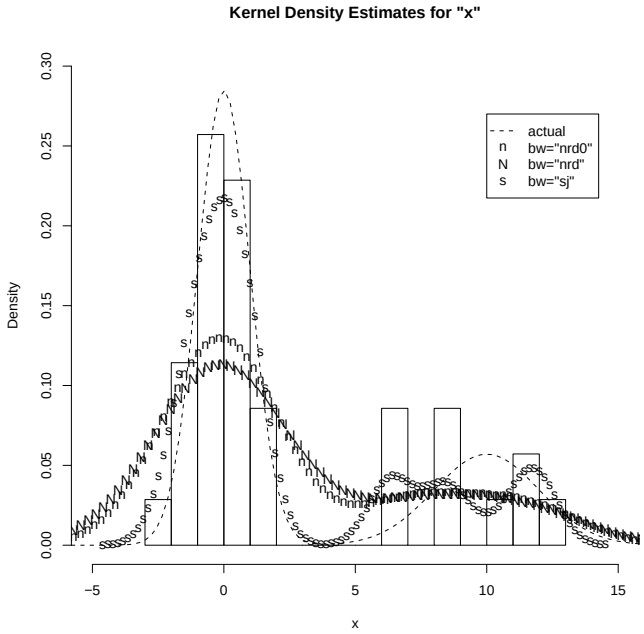
Default Bandwidth

The `nrd0`-chosen bandwidth works well with unimodal distributions and reasonable numbers of observations (here, $n = 50$). Solid is fitted, dashed is actual.





Sheather & Jones Bandwidth



Bootstrap Example #1

```
> x
 [1] 53 46 44 58 64 45 40 53 73 40 73 47 63 52 82
> (bs <- sample(x,replace=T))
 [1] 64 40 44 52 40 58 73 44 40 52 47 40 53 73 73
> mean(bs)
[1] 52.86667
> (bs <- sample(x,replace=T))
 [1] 53 73 52 44 73 64 53 40 45 53 82 45 40 64 45
> mean(bs)
[1] 55.06667
> bs.means <- numeric(0)
> for (i in 1:1000) {
+   bs <- sample(x,replace=T)
+   bs.means[i] <- mean(bs)
+ }
> mean(x)
[1] 55.53333
> mean(bs.means)-mean(x)
[1] 0.01766667
> sd(bs.means)
[1] 3.215488
> quantile(bs.means,probs=c(.05,.95))
      5%      95%
50.46667 61.13333
>
```

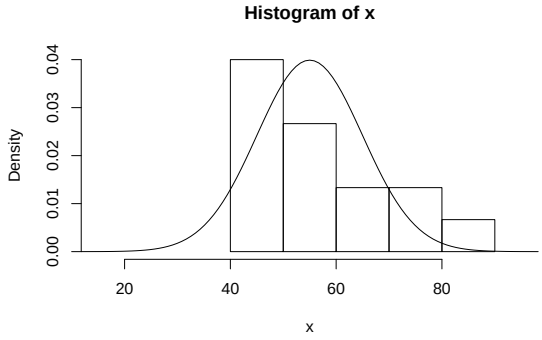
In fact, the sample was generated by:

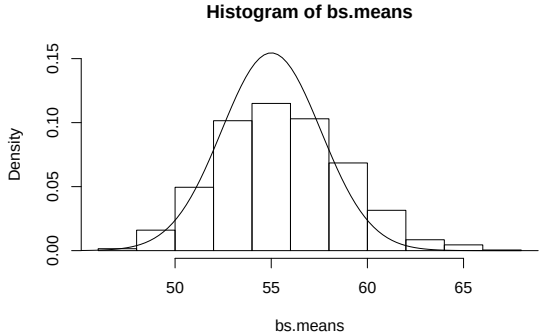
```
> x <- round(rnorm(15,mean=55,sd=10))
```

so $SE(\bar{X}) \approx \sigma/\sqrt{n} = 10/\sqrt{15} = 2.58$. Because the sample happened to have $s = 13.0$, our estimate of $SE(\bar{X})$ was closer to $13.0/\sqrt{15} = 3.36$.

Bootstrap Example #1

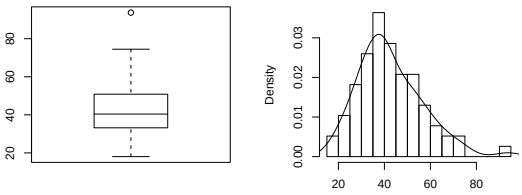
```
> hist(x, freq=F,xlim=c(15,95),breaks=5)
> curve(dnorm(x,55,10),add=T)
> hist(bs.means, freq=F,ylim=c(0,.15))
> curve(dnorm(x,55,10/sqrt(15)),add=T)
```





Bootstrap Example #2

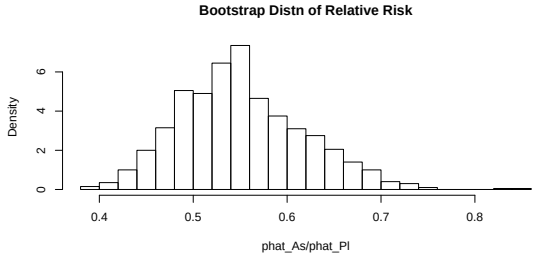
Cereal Rating:




```
> bs.means <- numeric(1000)
> bs.trimmeans <- numeric(1000)
> for (b in 1:1000) {
+   bs <- sample(rating,replace=T)
+   bs.means[b] <- mean(bs)
+   bs.trimmeans[b] <- mean(bs,trim=.1)
+ }
> mean(rating)
[1] 42.66570
> mean(bs.means)-mean(rating)
[1] 0.01660235
> sd(bs.means)
[1] 1.529426
> sd(rating)/sqrt(77)
[1] 1.600837
> mean(rating,trim=.1)
[1] 41.66178
> mean(bs.trimmeans)-mean(rating,trim=.1)
[1] 0.02696686
> mean(bs.trimmeans)-mean(rating)
[1] -0.976959
> sd(bs.trimmeans)
[1] 1.529365
>
```


Bootstrap Aspirin Example

```
> bs.as.heartattacks <- rbinom(1000,11037,104/11037)
> bs.pl.heartattacks <- rbinom(1000,11034,189/11034)
> phat.as <- bs.as.heartattacks/11037
> phat.pl <- bs.pl.heartattacks/11034
> relative.risk <- phat.as/phat.pl
> length(relative.risk)
[1] 1000
> (104/11037)/(189/11034)
[1] 0.550115
> mean(relative.risk)-.550115
[1] 0.001446324
> sd(relative.risk)
[1] 0.06653186  # compared to .06675 theoretical approx
> quantile(relative.risk, probs=c(.025,.975))
     2.5%     97.5%
0.4353592 0.6908454    # versus [0.416,0.684] theoretical approx
>
```



Jackknife for $\bar{X}$

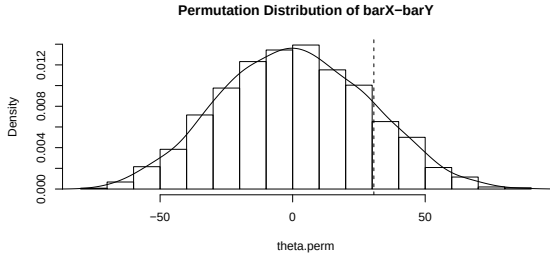
```
> x
 [1] 53 46 44 58 64 45 40 53 73 40 73 47 63 52 82
> theta.minusi <- numeric(0)
> n <- length(x)
> for (i in 1:n) {
+   theta.minusi[i] <- mean(x[-i])
+ }
> theta.minus <- mean(theta.minusi)
> theta <- mean(x)
> (n-1)*(theta.minus-theta)
[1] 0
> sqrt((n-1)/n*sum((theta.minusi-theta.minus)^2))
[1] 3.352137
> sd(x)/sqrt(n)
[1] 3.352137
>
```

Jackknife for $\bar{X}_\alpha$

```
> attach(cereal)
> theta.minusi <- numeric(0)
> n <- length(rating)
> for (i in 1:n) {
+   theta.minusi[i] <- mean(rating[-i],trim=.1)
+ }
> theta.minus <- mean(theta.minusi)  # *NOT* trimmed
> theta <- mean(rating,trim=.1)
> (n-1)*(theta.minus-theta)
[1] -0.5778958
> sqrt((n-1)/n*sum((theta.minusi-theta.minus)^2))
[1] 1.529543  # compared to SE_boot = 1.529365
>
```

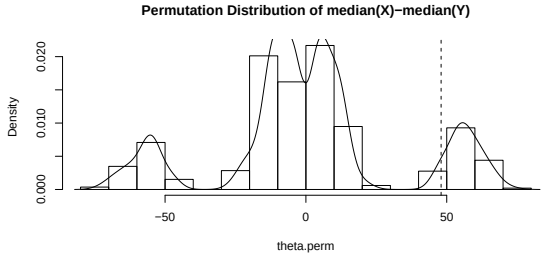

Permutation Test of Mice Means

```
> trt <- c(16,23,38,94,99,141,197)
> cntrl <- c(10,27,31,40,46,50,52,104,146)
> surv <- c(trt,cntrl)
> theta.perm <- numeric(2500)
> for (i in 1:2500) {
+   perm <- sample(surv)
+   trt.perm <- perm[1:7]
+   cntrl.perm <- perm[8:16]
+   theta.perm[i] <- mean(trt.perm)-mean(cntrl.perm)
+ }
> hist(theta.perm, main="Permutation Distribution of barX-barY", freq=F)
> lines(density(theta.perm, bw="sj"))
> (theta.observed <- mean(trt)-mean(cntrl))
[1] 30.63492
> abline(v=theta.observed, lty=2)
> sum(abs(theta.perm) > abs(theta.observed)
[1] 699
> sum(abs(theta.perm) > abs(theta.observed))/2500
[1] 0.2796
>
```



Permutation Test of Mice Medians

```
> theta.perm <- numeric(2500)
> for (i in 1:2500) {
+   perm <- sample(surv)
+   trt.perm <- perm[1:7]
+   cntrl.perm <- perm[8:16]
+   theta.perm[i] <- median(trt.perm)-median(cntrl.perm)
+ }
> hist(theta.perm, main="Permutation Distribution of median(X)-median(Y)",
       freq=F)
> lines(density(theta.perm))
> (theta.observed <- median(trt)-median(cntrl))
[1] 48
> sum(abs(theta.perm) > abs(theta.observed))
[1] 620
> sum(abs(theta.perm) > abs(theta.observed))/2500
[1] 0.248
>
```



Permutation Test of Paired Shoes

```
> theta.observed <- mean(A-B)
> theta.perm <- numeric(2500)
> for (i in 1:2500) {
+   flip <- (rbinom(10,1,.5)==1)
+   A.perm <- A
+   B.perm <- B
+   temp <- A.perm[flip]
+   A.perm[flip] <- B.perm[flip]
+   B.perm[flip] <- temp
+   theta.perm[i] <- mean(A.perm-B.perm)
+ }
> hist(theta.perm, main="Permutation Distribution of mean(A-B)", freq=F)
> lines(density(theta.perm))
> abline(v=theta.observed, lty=2)
> sum(abs(theta.perm)>abs(theta.observed))
[1] 20
> sum(abs(theta.perm)>abs(theta.observed))/2500
[1] 0.008
>
```