# Prostate Cancer Data

Load data

```
# training set
require(ElemStatLearn)
```

```
## Loading required package: ElemStatLearn
```

```
train <- data.frame( prostate[prostate$train,-10] )
test = data.frame( prostate[!prostate$train,-10] )
# all data?
# train <- data.frame( prostate[,-10] )
```

Settings

```
# Full model
fitF<-lm(lpsa ~ ., train)
X = model.matrix(fitF)
n = nrow(X)
p = ncol(X)
hatsigma2<-(summary(fitF)$sigma)^2
fit0 = lm(lpsa ~ 1, train)
xs<-names(train)[-9]
q = length(xs)
```

## Models

```
require(cherry)
```

```
## Loading required package: cherry
```

```
## Loading required package: bitops
```

```
## Loading required package: lpSolve
```

```
## Loading required package: Matrix
```

```
## Loading required package: slam
```

```
# converts integer to set notation
.num2names <- function(rejected, vars) {
  N <- length(vars)
  bools <- lapply(rejected, .bit2boolean, N=N)
  lapply(bools, function(b) vars[b])
}

# converts from integer to boolean (as binary)
.bit2boolean <- function(x, N) {
  base <- 2^(1:N-1)
  bitAnd(x, base) != 0
}

# models
Ms.temp = sapply(1:((2^q)-1), function(x) setdiff(xs,.num2names(x,xs)[[1]])  )
```

```
Ms = vector(mode="list", length=2^q)
Ms[[1]] = xs
for (i in 2:(2^q)) Ms[[i]] = Ms.temp[[i-1]]

# models size
ms = sapply(Ms,length) + 1

Mc = sapply(Ms,function(x) setdiff(xs,x))
```

### F test statistics and $p$-values

```
# F test
myFtest <- function(x) {
  others <- setdiff(xs, x)
  form <- formula(paste(c("lpsa~",  paste(c("1", others), collapse="+"))))
  anov <- anova(lm(form, data=train), fitF, test="F")
  res <- anov$"Pr("[2]                    # for R >= 2.14.0
  if (is.null(res)) res <- anov$"P("[2]   # earlier versions
  res
}

# raw p-values
ps = sapply(Mc,myFtest)
ps[1]= 1

# raw test stat
Fs = vector(mode="numeric", length=2^q)
Fs[2:2^q] = qf(ps[2:2^q], p-ms[2:2^q], n-p, lower.tail = FALSE)
Fs[1]=0
```

### Inferior models

```
# Inferior models
myalpha = 0.05
Infs = Fs*(p-ms) > q*qf(1-myalpha, q, n-p, ncp=q)
sum(Infs)
```

```
## [1] 117
```

```
pch_I = ifelse(Infs, 19, 1)
col_I = ifelse(Infs,  "red", "black")

require(beeswarm)
```

```
## Loading required package: beeswarm
```

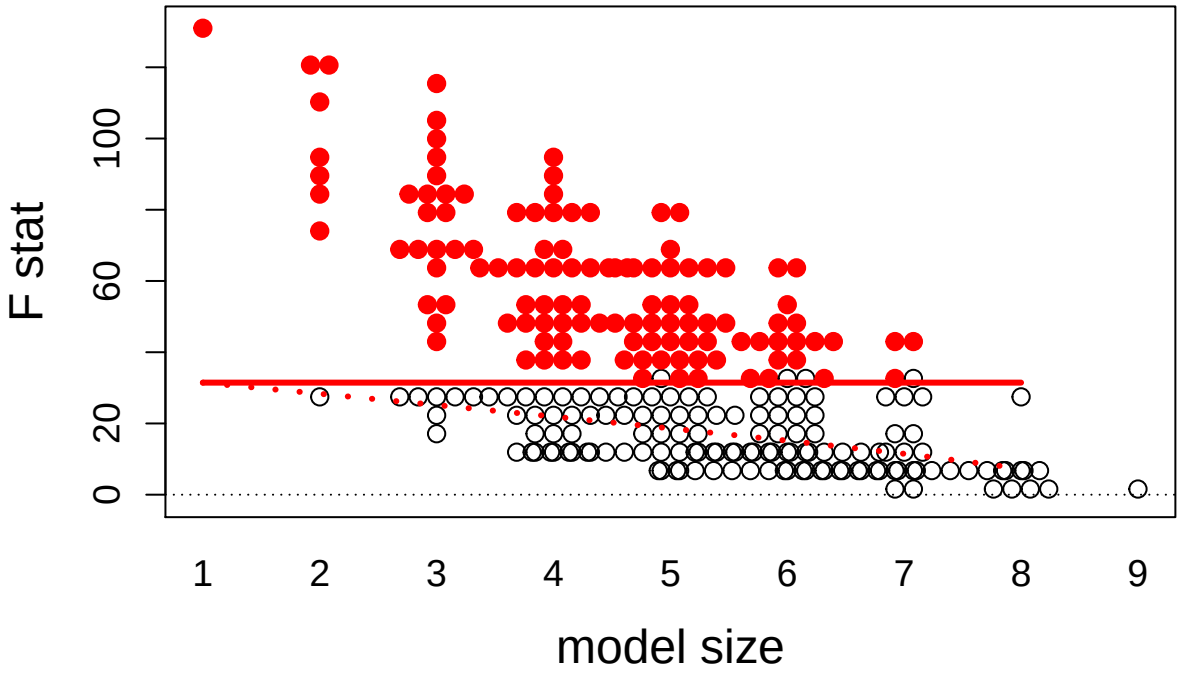
```
beeswarm(Fs*(p-ms) ~ ms,
         cex=1.2, cex.axis = 1.2, cex.lab=1.5,
         xlim=c(1,9),
         ylim=c(-1,max(Fs*(p-ms))),
         log=F,
         xlab="model size",
```

```
          ylab="F stat",
          pwpch=pch_I,
          method = c("center"),
          pwcol=col_I)

ub = (p-1:q)*qf(1-myalpha, p-(1:q),n-p, ncp=p-(1:q))
lines(1:q,ub,  lwd=3, col="red", lty=3)

sub = q*qf(1-myalpha, q,n-p, ncp=q)
lines(1:q,rep(sub,q), lwd=3, col="red", lty=1)

abline(h=0,lwd=1, lty=3)
```




### Predictions

```
# Predictions
```