Notes 1 Polynomials

```
fit <- lm(y ~ poly(x,d), train)
yhat <- predict(fit, newdata=test)
MSE.tr <- mean( (y - yhat)^2 )
fun <- function(d) if (d==0) lm(y~1, train) else lm(y~poly(x,d),
train)
fits <- sapply(ds, fun)
MSEs.tr <- unlist( lapply(fits, deviance) )/n
yhats <- lapply(fits, predict)
MSEs.te <- unlist(lapply(yhats, function(yhat) mean((y - yhat)^2)))
```

Notes 2 BiasVar

```
sim = function(d){
y = fx + rnorm(n,0,sigma)
fit = lm(y ~ poly(x,d))
yhat = fitted(fit)
}
yhats = replicate(B,sim(d))
Bias2 = ( fx - apply(yhats,1,mean) )^2
Var = apply(yhats,1,var)
Bias2 =  mean(
( fx - fitted( lm(fx ~ poly(x,d) ) ) )^2
)
Var = p*(sigma^2)/n
sim2 = function(d){
y = fx + rnorm(n,0,sigma)
fit = lm(y ~ poly(x,d))
yhat = fitted(fit)
ystar = fx + rnorm(n,0,sigma)
MSE.te = mean((ystar - yhat)^2)
}
```

Notes 3 ICCV

```
hatErrF = MSE.tr + (2*sigma^2*p)/n
hatsigma2 = (n*MSE.tr)/(n-p)
Cp = MSE.tr + (2*hatsigma2*p)/n
AIC <- AIC(fit)
BIC <- BIC(fit)
folds <- sample( rep(1:K, length=n) )
for (k in 1:K){
  fit <- lm(y~poly(x,d), train, subset=which(folds!=k))
  x.out <- train$x[which(folds==k)]
  yhat <- predict(fit, newdata=list(x=x.out))
  y.out <- train$y[which(folds==k)]
  KCV[k]<- mean( ( y.out - yhat )^2 )
}
KCV = cv.glm(train, glm(y~poly(x,d), train, family = gaussian), K=k)
$delta[1]
for (i in 1:n){
fit_i <- lm( y~poly(x,d), data=train[-i,])
yhat_i <- predict(fit_i, newdata=data.frame(x=train$x[i]) )
```

```
oneout[i] <- ( train$y[i] -  yhat_i )^2
}
X <- model.matrix(fit)
H <- X %*% solve(t(X)%*% X) %*% t(X)
mean(( (train$y - predict(fit)) / (1-diag(H))  )^2)
GCV = MSE.tr/(1-(p)/n )^2

# Notes 4 NPR

fit = kknn(y ~ x, train, test, kernel = "rectangular", k = k)
yhat = fit$fitted.values
LOOCV = train.kknn(y ~ x, train, ks = ks, kernel = "rectangular")
$best.parameters$k
MSE.tr = mean(
(train$y - kknn(y ~ x, train, test, kernel = "rectangular", k = k)
$fitted.values)^2
)
hatErr = MSE.tr + (2*sigma^2)/k
Bias2 = mean(
(ftrue - kknn(fx ~ x, train, test, kernel = "rectangular", k = k)
$fitted.values)^2
)
Var = sigma^2/k

# Notes 8 BAG

obs <- sapply(1:B,
              function(b)
              sample(1:n, size=n, replace=T)
              )
treelist <-lapply(1:B,
                  function(b)
                  rpart(fml, train[obs[,b],])
                  )
predict.bag <- function(treelist, newdata) {
  phats <- sapply(1:length(treelist),
                  function(b)
                  predict(treelist[[b]], newdata=newdata)[,"Class"]
                  )
  pbar <- rowMeans(phats)
}
phat2 = predict.bag(treelist, newdata=test)

# Notes 12 Ridge

hatbeta = solve(t(X)%*%X + lambda*diag(ncol(X))) %*% t(X) %*% y
```