1 Linear Regression

Formulas and Definitions

$\mathbf{Y} = \mathbf{X}\beta + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I}) \Rightarrow$

- $\hat\sigma^2 = \frac{1}{N-P}\Sigma_{i=1}^N (y_i - x_i^T\hat\beta)^2, \mathbf{E}(\hat\sigma^2) = \sigma^2$
- $\hat\beta = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$
- $\hat{\mathrm{Var}}(\beta) = \hat\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$
- $\hat\beta \sim \mathcal{N}(\beta, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1})$
- $\hat{\mathbf{Y}} \sim \mathcal{N}(\mathbf{X}\beta, \sigma^2\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top)$
- $\hat e = \mathbf{Y} - \hat{\mathbf{Y}} \sim \mathcal{N}(0, \sigma^2[\mathbf{I} - \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top)]$
- Projection matrix $\mathbf{H} = \mathbf{X}[\mathbf{X}\top\mathbf{X}]^{-1}\mathbf{X}\top$

RSS: residual sum square. $\Sigma_{i=1}(y_i - \hat y_i)^2$
LSR(Least square regression, minimize RSS): vertical distance, while **PCA**: perpendicular distance.
CI (confidence interval) $\hat\beta_j \pm \hat{se}(\hat\beta_j)t_{1-\frac{\alpha}{2},n-p}$, suitable for: estimated parameters β_j, the independent variable y
PI (prediction interval) $\hat\beta_j \pm \hat{se}(\hat\beta_j)t_{1-\frac{\alpha}{2},n-p} + \hat\sigma^2$, suitable only for y
CI: $\frac{\mathbf{x}_0^\top\hat\beta - \mathrm{E}[\mathbf{y}_0]}{\sigma\sqrt{\mathbf{x}_0^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_0}} \sim t_{n-p}$ PI: $\frac{\mathbf{x}_0^\top\hat\beta - \mathbf{y}_0}{\sigma\sqrt{\mathbf{x}_0^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_0+1}} \sim t_{n-p}$
KNN $k \uparrow \Rightarrow$ Bias $\uparrow$, Var $\downarrow$, Smoothness $\uparrow$, (generally) estimated TestMSE $\downarrow$
Curse of dimension KNN is sensible to the dimensions because of the neighbor definition based on norm. It may be avoided by a suitable neighbor definition.
Good-of-fit $\hat\sigma^2$: smaller better. small$\rightarrow$ much explained by the model. one-σ: 2/3, two-σ: 95%.
$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum_{i=1}(y_i-\hat y_i)^2}{\sum_{i=1}(y_i-\bar y)^2}$, bigger better.

$$\max R^2 \Leftrightarrow \min RSS \Leftrightarrow LSR$$

adjusted$R^2 = 1 - \frac{RSS/(n-p)}{TSS/(n-1)}$, used to compare models with **different** df. Should be positive but not necessarily.
Power: the probability of rejecting H_0 when H_α is true.
F-statistic $F = \frac{RSS_1/df_1}{RSS_2/df_2}$

Practical

Plot: Residual VS Fitted (Tukey-Anscombe plot): to test if $E(\varepsilon) = 0$ (the red line is horizontal at v=0)
Plot: QQ: to test if following normal distribution. (A diagonal straight line)
Plot: Cooks' distance: high value means deleting this point would change the model a lot.
Plot: Leverage: high means this point is far away from others. $h_i = [\mathbf{H}]_{ii} = \frac{\partial \hat y_i}{\partial y_i}$
An observational study cannot be used to generate casual conclusions.

```
XtX.inv<-solve(crossprod(X)) # Make sure
    that first column of X are 1's)
beta.hat<-XtX.inv%*%t(X)%*%y # estimates
y.hat<-X %*% beta.hat # fitted values, or
residuals(fit)
res<-y-y.hat # residuals
RSE<-sqrt(sum(res^2)/(n-p)) # estimate res
RSS<-sum(res^2)
TSS<-sum((y-mean(y))^2)
Rsquared<-1-RSS/TSS
Rsquared.adj<-1-(RSS/(n-p))/(TSS/(n-1))
se.beta.i<-RSE*sqrt(XtX.inv[i,i])
t_i<-beta.hat[i] / se.beta.i # t-value
```

```
coef<-summary(fit1)$coefficients #
    alternative t_value and se
se1<-coef["x1","Std. Error"]
beta1<-coef["x1","Estimate"]
```

```
t1<-beta1/se1
p.val <- 2*pt(abs(t_i),df=n-p, lower=F)
p.val.alt <- 2*pt(-abs(t_i),df=n-p)
fit.TV.radio <- lm(sales ~ TV + radio,
    data=Advertising)
anova(fit.TV.radio, fit.all)} #compare two
    models with and without variable
fit.empty <- lm(y ~ 1, data=x.frame)
anova(fit.empty,fit.all)} # F-test
```

```
Ftest.alt <- summary(fit1)$fstatistic
f.p.val <- 1 - pf(Ftest.alt[1], df1 =
    Ftest.alt[2], df2 = Ftest.alt[3])

plot(fit,which =1) # residual vs fitted
plot(fit,which =2) # normal QQ
plot(fit,which =4) # Cook's distance
plot(fit,which =5) # Leverage

glm.fit = glm(nox ~ poly(dis, i), data =
    Boston) # glm has cv function
```

```
cv.glm(Boston, glm.fit, K = 10)$delta[1] #
    estimated predict err

# CI for beta and y0
confint(fit) # fit: an lm object
n <- nrow(dat) # by hand, for beta1, 95%
m <-ncol(dat)
coef(fit)[1] - qt(.975, n-1-m)*sigma
coef(fit)[1] + qt(.975, n-1-m)*sigma
quant <- qt(.975,n-1-m)) # for y0, 95%
```

```
sigma.hat <- sqrt(sum((fit$resid)^2)/(n-1-
    m))
X <- as.matrix(cbind(1,x0))
XtXi <- solve(t(X) %*% X)
x00 <- as.matrix(c(1,x0),nrow=m+1)
se <- sigma.hat * sqrt( t(x00) %*% XtXi %*
    % x00)
# fitted +(-) quant*se # by hand
predict(fit, pred.frame, level=.95,
    interval="c")#make sure colname(pred.
    frame)== colname(data.frame(..)) used
    in lm!!
# PI: for y0
se.pi <- sigma.hat * sqrt( 1 + t(x00) %*%
    XtXi %*% x00)
# fitted +(-) quant*se.pi # by hand
predict(fit, pred.frame, level=.95,
    interval="p")#automatically
preds=predict(fit,newdata=newx, se=TRUE)
```

```
se.bands=cbind(preds$fit+2*preds$se.fit,
    preds$fit-2*preds$se.fit)
```

```
# KNN
kn<-knn.reg(train = matrix(dtrain[,1],ncol
    =1),y = ytrain, test = matrix(dtest
    [,1],ncol=1), k = k) #If X is 1-D,
    need to be formed as matrix of df.
kn$pred # get the prediction of KNN model
```

2 Bias-Variance and Cross Validation

Bias-Variance

Decomposition $\mathrm{E}_\Omega[(y - \hat f(x))^2] = [f(x) - \mathrm{E}_\Omega \hat f(x)]^2 + \mathrm{E}_\Omega[(\mathrm{E}_X[\hat f(x)] - \hat f(x))^2] + \sigma^2, \Omega = \{X\}$
- Bias$^2 = [f(x) - \mathrm{E}_\Omega \hat f(x)]^2$: the model $\hat f(\cdot)$'s bias compared to all the models.
- Variance $= \mathrm{E}_\Omega[(\mathrm{E}_X[\hat f(x)] - \hat f(x))^2]$: the variance among models trained on different datasets.
- randomness σ^2

Flexibility $\uparrow \Rightarrow$ Training RSS $\downarrow$, Var $\uparrow$, Bias $\downarrow$, random error (irreducible error) $\rightarrow \Rightarrow$ test RSS: first $\downarrow$, then $\uparrow$

CV

LOOCV $\hat y_i^{(-i)} = \frac{(\mathbf{HY})_i - \mathbf{H}_{ii}y_i}{1-\mathbf{H}_{ii}}, \mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$
Double CV outer: model assessment; inner: model selection. The inner dataset is part of the outer's.
Comparison:
- Randomness: LOOCV: No, K-fold: yes, from the randomness of splitting the data. Valid: yes, large
- Computation cost: LOOCV: high(low using formula), K-fold: lower, Valid: much low
- Estimated test MSE:
LOOCV: $\hat\theta_{LOOCV} = \frac{1}{n}\sum_i(y_i - \hat f^{(i)}(x_i))^2$
k-fold: $\hat\theta_{\text{k-fold}} = \frac{1}{K}\sum_{j=1}^K \frac{1}{|I_j|}\sum_{i\in I_j}(y_i - \hat f^{(j)}(x_i))^2$
Valid: $\hat\theta_{\text{Valid}} = \frac{1}{|\text{Valid}|}\sum_{i\in\text{Valid}}(y_i - \hat f(x_i))^2$
- Bias and variance: no clear clue.

```
sample(cut(1:nrow(dat), breaks = 10,
    labels = F),nrow(dat), replace=F) #
    randomly cut dataset into 10-folds

# Bias-Variance trade-off
ExpTestMSE <- apply((fit.test-y.test)^2,2,
    mean) #fit.test: nsim*nmodels
Bias2<-(apply(fit.test,2,mean)-f(xtest))^2
Var <- apply(fit.test,2,var)
VarEps <- var(y.test)
Bias2+Var+VarEps - ExpTestMSE
```

3 Bootstrap

Consistency

Definition

$P(a_n(\hat\theta_n - \theta) \le x) - P^*(a_n(\hat\theta_n^* - \hat\theta_n) \le x) \xrightarrow{P} 0$ as $n \to \infty$

Usage
- $\mathrm{Bias}(\hat\theta_n) := \mathrm{E}[\hat\theta_n] - \theta \approx \mathrm{E}^*[\hat\theta_n^*] - \hat\theta_n \approx \frac{1}{B}\sum_{b=1}^B \hat\theta_n^{*b} - \hat\theta_n$
- $\mathrm{Var}[\hat\theta_n] \approx \mathrm{Var}^*[\hat\theta_n^*] \approx \frac{1}{B-1}\sum_{b=1}^B(\hat\theta_n^{*b} - \overline{\hat\theta_n^*})^2$ The first $\approx$ requires n large enough, the second asks for B.

Hold Condition $\sqrt{n}(\hat\theta_n - \theta)$ is asymptotically normal.

CI Types

Quantile Bootstrap CI (Naive CI)
directly use the quantile of $\hat\theta_n^*$.
$[q_{\hat\theta_n^*}(\frac{\alpha}{2}), q_{\hat\theta_n^*}(1-\frac{\alpha}{2})]$
Reversed quantile CI(Pivotal CI):
using $q_{\hat\theta_n^* - \hat\theta_n}$ to replace $q_{\hat\theta_n - \theta}$
$[\hat\theta_n - q_{\hat\theta_n^* - \hat\theta_n}(1-\frac{\alpha}{2}), \hat\theta_n - q_{\hat\theta_n^* - \hat\theta_n}(\frac{\alpha}{2})]$
Normal Bootstrap CI:
using $z \sim \mathcal{N}(0,1)$ for quantile and $\mathrm{Var}(\hat\theta_n^*)$ for variance
$[\hat\theta_n - q_z(1-\frac{\alpha}{2})\sqrt{\mathrm{Var}(\hat\theta_n^*)}, \hat\theta_n + q_z(1-\frac{\alpha}{2})\sqrt{\mathrm{Var}(\hat\theta_n^*)}]$
Bootstrap T: best accurate $O(\frac{1}{n})$, compute intensively.
using t-like statistics $t = \frac{\hat\theta_n - \theta}{\hat{\mathrm{sd}}(\hat\theta_n)} \leftarrow t^* = \frac{\hat\theta_n^* - \hat\theta_n}{\hat{\mathrm{sd}}(\hat\theta_n^*)}$
$[\hat\theta_n - q_{t^*}(1-\frac{\alpha}{2})\hat{\mathrm{sd}}(\hat\theta_n), \hat\theta_n + q_{t^*}(\frac{\alpha}{2})\hat{\mathrm{sd}}(\hat\theta_n)]$
- $\hat{\mathrm{sd}}(\hat\theta_n^{*b}) = \sqrt{\frac{1}{C-1}\sum_{c=1}^C(\hat\theta_n^{**bc} - \overline{\hat\theta_n}^{**bc})^2}$
- $\hat{\mathrm{sd}}(\hat\theta_n) = \sqrt{\frac{1}{B-1}\sum_{b=1}^B(\hat\theta_n^{*b} - \overline{\hat\theta_n}^{*b})^2}$

```
# Bootstrap
fun.theta <- function(x, ind) {...}
fun.gen <- function(x, mle) {...}
boot.parametric <- boot(dat, statistic =
    fun.theta, R = nr_bootsamples, sim =
    "parametric", ran.gen = fun.gen, mle
    =...,...)
boot.nonparametric <- boot(dat, statistic=
    fun.theta, R=nr_bootsamples)
boot.ci(boot.out = boot.parametric, type =
    c("norm", "basic", "perc"), index=1)
    #basic: quantile, perc: reversed
    quantile. index: choose if boot
    objects has two or more theta.
```

```
theta.hat<-res.boot$t0
```

```
theta.hat.star<-res.boot$t
plot(boot.parametric, index = 1)
# CI by hand
quantile.CI <- quantile(bootstrap.theta,
    probs = c(0.025, 0.975))
normal.CI <- c(original.theta - qnorm
    (0.975) * sd(bootstrap.theta),
    original.theta + qnorm(0.975) * sd(
    bootstrap.theta))
reversed.CI <- original.theta - quantile(
    bootstrap.theta - original.theta,
    probs = c(0.975, 0.025))
```

4 Testing

Permutation test

Assumptions:
1. The two samples are independent of one another
2. The two populations have equal variance or spread
3. The two populations are normally distributed

Parametric test: z-test: known variance; t-test: unknown
Non-parametric test: Permutation(Randomization) test.
- Special case: Wilcoxon rank sum test (Mann-Witney U test). No assumption 3.

$H_0: F_1 = F_2$, H_A:F_1 is a shifted version of F_2
rank on all the datasets; compute the rank sum within each group; compare.
- permutation p-value: the more extreme ratio than the observed among all the permutation datasets.
- for multiple variables: **cannot** permute single column, cannot directlt test individual coefficients.

Multiple testing

	H_0 is true	H_a is true	Total
H_0 is not rejected	U	T	$m-R$
H_0 is rejected	V	S	R
Total	m_0	$m-m_0$	m

Error measurements:
FWER: family-wise error rate: $P(V \ge 1)$
FDR: false-discovery rate: $\mathrm{E}[Q], Q = \frac{V}{R}$
FDR $\le$ FWER. $\alpha \le$ FWER $\le m\alpha$
Control methods:
FWER: Bonferroni (Intuitive) control, Westfall-Young permutation procedure
FDR: Benjamini-Hochberg.
Bonferroni control: choose $\alpha = \frac{\alpha^*}{m}$ for each hypotheses testing.
Too strict. Unsuitable for: (1)m very large. Because FWER$\le m\alpha - \frac{m(m-1)}{2}\alpha^2 + O(\alpha^2)$ or (2) hypothesis are dependent. **Westfall-Young** choose δ, s.t. FWER $= P(V \ge 1) = P(\min\{p_1, p_2, \cdots, p_m\} \le \delta) \le \alpha \Rightarrow$ find the α-quantile of $D := \min\{p_i\}$
Benjamini-Hochberg theory: controls FDR $= \frac{m_0}{m}q \le q$.
1. ascending order of p-values: $p_{(1)} \le p_{(2)} \le \cdots \le p_{(m)}$.
2. given q, $i_0: \arg_i \min p_j \ge \frac{j}{m}, \forall j > i$
3. reject all the first i_0 hypotheses.

```
# Wilcoxon test:
#   H0: F_1 = F_2
#   Ha: F_1 is shifted to the left
wilcox.test(Y1,Y2, alternative="less") # unpaired
wilcox.test(Y1-Y2, alternative = "greater") # paired
wilcox.test(Y1,Y2, alternative = "greater", paired = TRUE) # equivalent

# By hand
Wilxocon.one.permutation <- function(y){
  n <- length(y)
  signs <- sample(c(-1, 1), n, replace = TRUE)
  d <- y * signs
  d.rank.sign <- rank(abs(d)) * sign(d)
  ranks.pos <- sum(d.rank.sign[d.rank.sign > 0])
  return(ranks.pos)
}
dd <- Y1 - Y2
dd <- dd[dd != 0]
res <- replicate(100000, Wilxocon.one.permutation(dd))
```

5 Feature Selection

Judge criterion

Mallow's C_p $\frac{1}{n}(RSS+2d\hat{\sigma}^2)$

AIC $\frac{1}{n\hat{\sigma}^2}(RSS+2d\hat{\sigma}^2)=\frac{C_p}{\hat{\sigma}^2}=\frac{RSS}{n\hat{\sigma}^2}+\frac{2d}{n}$

BIC $\frac{RSS}{n\hat{\sigma}^2}+\frac{d}{n}\log n$

Adjusted R-sq $1-\frac{RSS/(n-d-1)}{TSS/(n-1)}$

Subsets construction

Exhaustive k given: $\binom{p}{k}$ times; not given: 2^p times. **Forward step-wise**: from the empty model, each step add one new variable. **Backward**: from the full model, each step drop one variable.

Norm shrinkage

Ridge $\arg_{\beta:\beta\in\mathbb{R}^p}\min \mathrm{RSS}(\beta)+\lambda_s\|\beta\|_2 \Rightarrow \hat{\beta}^{\mathrm{Ridge}}=(\mathbf{X}^\top\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{Y}$
Coefficients tend to be similar. Because the objective restricts $\lambda\|\beta\|_2$. $\hat{\beta}_i=\hat{\beta}_j=\frac{c}{2}\arg_{\beta_i+\beta_j=c}\min(\beta_i^2+\beta_j^2)$ is the smallest.
LASSO restrict 1-norm. **Adaptive LASSO** $\hat{\beta}_s=\arg_\beta\min RSS(\beta)+\lambda\sum_{j=1}^n w_j|\beta_j|$

Group LASSO $\hat{\beta}_\lambda=\arg_{\beta\in\mathbb{R}^p}RSS(\beta)+\lambda\sum_{l=1}^K w_l\|\beta_l\|_1$
Highly correlated variables LASSO: choose one of them (can be very unstable); Ridge: divide the weights among them.
Practical procedure using LASSO/Ridge, first center and standarize data. **Orthogonality** adding(removing) variables does not change the fitted values.
Because $\mathbf{X}^\top\mathbf{X}=\mathrm{Diag}(\{\sum_{i=1}^n x_{i1}^2\})\Rightarrow$
$\hat{\beta}_j=\frac{\sum_{i=1}^n x_{ij}y_i}{\sum_{i=1}^n x_{ij}^2}, se(\hat{\beta}_j)=\sqrt{\hat{\sigma}^2/\sum_{i=1}^n x_{ij}^2}$

```
# subset selection
regfit<-regsubsets(Salary~.,data=train[folds!=i,], nvmax=19, method="forward") # default is "exhaustive"
reg.summary=summary(regfit)
reg.summary$rsq # also adjr2, cp, bic
bic_id<-which.min(reg.summary$bic)
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}

# LASSO and Ridge
ridge.model<-glmnet(x = data$x,y = data$y, alpha = 0,lambda=grid, thresh=1e-12)
    # alpha 0:ridge, 1:lasso
predict(ridge.model, s=4, newx=x[test,])
predict(ridge.model,type="coefficients",s=4) # get the predicted coefficients
plot(ridge.model)
cv.ridge=cv.glmnet(x[train,],y[train], alpha=0)
bestlam=cv.out$lambda.min
```

6 Splines

Degreee-d spline

Guarantee the $d-1$ degree continuity at the split points. **df** $d+k+1$

Natural cubic spline

df (linear outside the outer knots): $k=3+k+1-(3-1)*2$
A spline with higher degree of freedom will **not** always make systematically the training error lower.

Smoothing

trade-off between SSE and smoothness.
$\hat{g}=\arg_g\min\sum_{i=1}^n(y_i-g(x_i))^2+\lambda\int_a^b g''(x)^2dx$
$\lambda\to 0$: Degree of freedom is n
$\lambda\to\infty$: $g''(x)=0$, the LSR linear estimation. Df: 2

Local

$\hat{\beta}_i=\arg_{\beta_i}\min\sum_{j=1}^n w_{ij}(y_j-x_j^\top\beta_i)^2$
poorly for high-dimension (larger than 3 dimensions) ; unsupervised and non-parametric.

```
step.model <- step(lm(y ~ ., data=dat), k = log(n), trace = 0) # k=2: AIC, log(n): BIC
poly(x,power) # construct orthogonal basis

fit=lm(y~bs(x,knots=c(x1,x2,x3)),data=dat)

# natural spline
attr(ns(age,df=4),"knots") # 3 knots-> 4 dimension data. df=K+1
attr(ns(age,df=4, intercept = T),"knots")
    # intercept=T -> add another column, intercept, in the data matrix. df=K+2

# local spline
loess(y~x,span=.2,data=dat)

# GAM
gm <- gam(form.gam, data = dat) # GAM model e.g. form.gam: y~s(X1,4)+s(X2,3)
```

7 Tree-based models

Definition $f(x)=\sum_{m=1}^M c_m * 1_{(X\in R_m)}$
Properties:
- equal to fit a linear model on $\{1_{(X\in R_m)}\}$
- covariates $\{1_{(X\in R_m)}\}$ are all orthogonal.

Recursive binary splitting: repeat until convergence.
1. find the best cutting point for each predictor
2. $\hat{y}_{R_1},\hat{y}_{R_2}=\arg_{y_{R_1},y_{R_2}}\min\sum_{i:x\in R_1}(y_i-\hat{y}_{R_1})^2+\sum_{i:x\in R_2}(y_i-\hat{y}_{R_2})^2)$

Pruning the tree not necessarily improve the test MSE.
$\min_T\sum_{m=1}^{|T|}\sum_{i:x_i\in R_m}(y_i-\hat{y}_{R_m})^2+\alpha|T|$

Bagging

bootstrap multiple datasets and train tree on each dataset.
$\hat{f}_{bag}(x)=\frac{1}{B}\sum_{b=1}^B\hat{f}^{*b}(x)$.

Random Forest

bagging datasets, subset variables (features).

```
tree.model = tree(Y~x, dat, subset=train)
    # train a tree model
predict(tree.model, dat.test, type="class") # classification
predict(tree.boston, dat.test) # regression
plot(tree.model) # plot tree
text(tree.model,pretty=0) # add labels

# prune the tree
prune.model=prune.tree(tree.model,best=5)
prune.model.class <- prune.misclass(tree.model, best=7) #for class.

# CV
cv.tree(tree.reg.model)
cv.tree(tree.class.model, FUN=prune.misclass)

# random forest
rf.model=randomForest(y~.,data=dat, subset=train1,mtry=13,importance=TRUE) # mtry: number of features sampled in each split
rf.class.model <- randomForest( as.factor(y) ~ ., data=dat, mtry=3, norm.votes=TRUE, maxnodes=5)

varImpPlot(rf.model)
```

8 General Hints

Technical words

RSS, MSE

Theories

1. the distribution of p-value is uniform under the null hypothesis.
Proof Test statistic T has the distribution $F(T)$. P: the p-value p of a given t is defined as $p=f(F,t)=\Pr(F(T)\le t)=F(t)\Rightarrow P=F(T)\Rightarrow\Pr(P<\alpha)=\Pr(F(T)<\alpha)\equiv\alpha$ (the definition of p-value of T).
2. Curse of dimensions: the points concentrate on the sphere when the number of dimensions increases. This means for example that if we have in mind to estimate a regression function near 0, then few points will be sufficiently close to it in high-dimensions.

```
dat<-na.omit(dat) #remove entries with NA
cut(1:100, breaks = 10,labels = F) # cut into 10 folds.
which.min(aList) # return the id of the minimum value in aList ( a list).
as.formula(paste("y~s(", paste(list.var, collapse = ", 4)+s(")
        ,",4)", sep = "")) # construct formula
fit.gamma <-fitdistr(dat, "gamma") #fit a parametric distr

# Batch process
ExpTestMSE <- apply((fit.test-y.test)^2,2, mean) # 2:means working on column.
mapply
sapply
replicate(t, func(...)) # perform the func t times

# distribution related
ecdf(array) # fit empirical cumulative distribution of array
density(array) # fit the density. default: Gaussian
dnorm(x,mean,sd) # densityfunction f(x)
pnorm(x,mean,sd) # cdf: p(X<=x)
# generate new random numbers
rnorm(n,mean,sd) # normal
runif(n,min,max) # uniform
rexp(n, rate) # exponential

rank(v) # return the rank
order(v) # return the index

combn(array,n) # the combination of taking n items from array.

round(x, 4) # digits = 4
```

Packages and functions
boot boot
gam gam
glmnet glmnet
kknn
leaps regsubsets,
randomForest
splines
tree tree, cv.tree