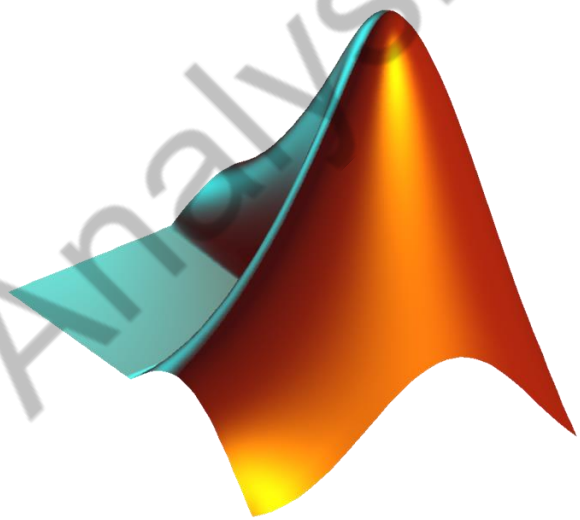
DATA ANALYSIS

MATLAB



CHAPTER 2

Distributions

```
X = exprnd(tau,n,M);
```

```
X = unifrnd(a,b,n,M);
```

```
X = possrnd(lamda,n,M);
```

```
X = rand(n,M);
```

```
mean_matrix = [muX muY];

sigmaXY = rho*sigmaX*sigmaY;

covariance_matrix = [sigmaX^2 sigmaXY; sigmaaXY
sigmaY^2];

data = mvnrnd(mean_matrix, covariance_matrix, n);

X = data(:,1);

Y = data(:,2);
```

Histogram Techniques 1

```
[Nx,Xx] = hist(X,bins);

ypdf = tau*exp(-tau*Xx); %for poisson

ypdf = ypdf/sum(ypdf);

figure(1)

plot(Xx,Nx/n,'.-k')

hold on

plot(Xx,ypdf,'c')

legend('simulated','analytic')

figure(2)

histfit(X,bins,'exponential')

%or
```

```
width = Xx(2) - Xx(1);
relfreq = Nx/n;
estypdf = relfreq/width;
fexppdf = @(x) tau*exp(-tau*x);
figure(3)
plot(Xx,estypdf_,'-k')
hold on
fplot(fexppdf,[Xx(1)-width/2 Xx(end)+width/2],'c')
legend('simulated','analytic')
```

Histogram Techniques 2

```
[counts, centers] = hist(X,bins);
figure(1)
bar(centers,counts)
hold on
plot([mean(X) mean(X)],[0 1.1*max(counts)],'r')

normcdf(value, mu, sigma);
norminv(value, mu, sigma);
```

CHAPTER 3

```
mean(X);
var(X);
std(X);
median(X);
mle(X,'distribution','poisson');
```

```
custom_pdf = @(data, lamda)lamda*exp(-lamda*data);
custom_cdf = @(data,lamda)1-exp(-lamda*data);
phat = mle(data,'pdf', custom_pdf, 'cdf', custom_cdf);
```

```
tinv(1-alpha/2, n-1);
tcdf(1-alpha/2, n-1);
```

Parametric Hypothesis Testing & CI for mean μ_x

```
[H, P, CI] = ttest(X, test_value, alpha);
```

Parametric Hypothesis Testing & CI for variance

```
[H, P, CI] = vartest(X, test_value, alpha);
```

Goodness-of-fit test

```
[H, P, STATS] = chi2gof(X,'cdf',@(z)normcdf(z,mean(X),
std(X)), 'nparams',2);
for i=1:length(STATS.O)
     fprintf('\t %3.3f \t %3.3f \n',STATS.O(i),
STATS.E(i));
end
```

Create B bootstrap samples of mean/variance

```
bootmu = bootstrp(B,@mean,X);
se = std(bootmu)
```

```
bootvar = bootstrp(B,@var,X);
```

Bootstrap CI of mean/variance

```
CI = bootci(B,{@mean,X},'alpha',alpha);
CI = bootci(B,{@var,X},'alpha',alpha);
```

Parametric CI & Hypothesis Testing for difference of means $\mu_x - \mu_y$

```
[H,P,CI] = ttest2(X,Y,alpha);
```

Bootstrap CI for difference of means $\mu_x - \mu_y$

```
CI = NaN*ones(2,M);
for i=1:1:M
    bootstatX = bootstrp(B, @mean, X(:,i));
    bootstatY = bootstrp(B, @mean, Y(:,i));
    bootstatXY = bootstatX - bootstatY;
    bootstatXY = sort(bootstatXY);
    k = floor((B+1)*alpha/2);
    CI(1,i) = bootstatXY(k);
    CI(2,i) = bootstatXY(B+1-k);
end
```

Bootstrap Hypothesis Testing/Randomization Hypothesis Testing for difference of means $\mu_x - \mu_y$

```
replacement = true/false;

XY = [X; Y];

bootstrapxy =  NaN*ones(1,B);

    for k=1:1:B

        bootstrap_samples =
randsample(XY,m+n,replacement);

        x_mean = mean(bootstrap_samples(1:n));

        y_mean = mean(bootstrap_samples(n+1:n+m));

        bootstrapxy(k) = x_mean - y_mean;

    end

   bootstrapxy = sort(bootstrapxy);

   [~,pos] = min(abs(bootstrapxy-(mean(X(:,j))-
mean(Y(:,j)))));


   if(pos < ( (B+1)*alpha )/2 | pos>(B+1)*(1-alpha/2))

       H = 1; %reject null hypothesis

   else

       H = 0; %accept null hypothesis

   end
```

CHAPTER 5

[5.1] PEARSON CORRELATION COEFFICIENT

Find Pearson Correlation Coefficient (rtest.m)

```
temp = corrcoef(X,Y);
r = temp(1,2);
%or
temp = cov(X,Y);
r = temp(1,2);
%or
r = corr(X,Y);
```

CI of r (rtest.m)

```
z = 0.5*log((1+r)./(1-r));
z_low = z-(norminv(1-alpha/2)*sqrt(1/(n-3)));
z_upper = z+(norminv(1-alpha/2)*sqrt(1/(n-3)));
r_low = tanh(z_low);
r_upper = tanh(z_upper);
CI = [r_low r_upper];
```

Hypothesis Testing for r (rtest.m)

```
t0 = abs(r.*sqrt((n-2)./(1-r.^2)));
if t0<-tinv(1-alpha/2,n-2) | t0>tinv(1-alpha/2,n-2)
        H = 1;
else
        H = 0;
end
```

Randomization Hypothesis Testing for r

```
X; Y;
r = corr(X,Y);
t0 = r*sqrt((n-2)/(1-r^2));
t = NaN*ones(L,1);
for j=1:L
        X2 = randsample(X,n,false);
        rr = corr(X2,Y);
        t(j) = rr*sqrt((n-2)*(1-rr^2));
    end
    t = sort(t);
    [~,pos] = min(abs(t-t0));
    if pos<L*alpha/2 | pos>L*(1-alpha/2)
        H = 1;
    else
        H = 0;
    end
end
```

[5.2] LINEAR REGRESSION

Linear Regression: Find b (btest.m)

```
x; y;
X = [ones(length(x),1) x];
b = X\y;
yfit = X*b;
e = y - yfit;
se = std(e);
figure(1)
scatter(x,y)
title('Scatter Plot')
```

```
hold on
plot(x,yfit,'r')
legend('Data','y = b_0 + b_1*x','Location','best');
```

Parametric CI of b (btest.m)

```
x; y;
X = [ones(length(x),1) x];
b = X\y;
sxx = (n-1)*var(x);
sb0 = se*sqrt(1/n + mean(x)^2/sxx);
b0_low = b(1) - tinv(1-alpha/2,n-2)*sb0;
b0_upper = b(1) + tinv(1-alpha/2,n-2)*sb0;
sb1 = se/sqrt(sxx);
b1_low = b(2) - tinv(1-alpha/2,n-2)*sb1;
b1_upper = b(2) + tinv(1-alpha/2,n-2)*sb1;
bCIparametric = [b0_low b0_upper;b1_low b1_upper];
```

Future and Mean prediction

```
hold on
meanx = mean(x);
Xstep = min(x):0.01:max(x);
plot(Xstep,(b0+b1*Xstep)-tinv(1-alpha/2,n-
2)*se*sqrt(1/n+((Xstep-meanx).^2)/sxx),'-g')
hold on
plot(Xstep,(b0+b1*Xstep)+tinv(1-alpha/2,n-
2)*se*sqrt(1/n+((Xstep-meanx).^2)/sxx),'-g')
hold on
plot(Xstep,(b0+b1*Xstep)-tinv(1-alpha/2,n-
2)*se*sqrt(1+1/n+((Xstep-meanx).^2)/sxx),'-m')
hold on
plot(Xstep,(b0+b1*Xstep)+tinv(1-alpha/2,n-
2)*se*sqrt(1+1/n+((Xstep-meanx).^2)/sxx),'-m')
```

Bootstrap CI of b (btest.m)

```
x; y;

n = length(X);

M = 1000;

b0_array = NaN*ones(M,1);
b1_array = NaN*ones(M,1);

for i=1:1:M
    rnd = unidrnd(n,n,1);
    Xboot = x(rnd);
    Yboot = y(rnd);

    XX = [ones(n,1) Xboot];
    bboot = XX\Yboot;
    b0_array(i) = bboot(1);
    b1_array(i) = bboot(2);
end

b0_array = sort(b0_array);
b1_array = sort(b1_array);

b0_low = b0_array(floor(M*alpha/2));
b0_upper = b0_array(floor(M*(1-alpha/2)));

b1_low = b1_array(floor(M*alpha/2));
b1_upper = b1_array(floor(M*(1-alpha/2)));

bCIbootstrap = [b0_low b0_upper;b1_low b1_upper];
```

Diagnostic plot

```
figure(1)
e_star = e/std(e);
plot(e_star,'o');
hold on
xx = -1:n+1;
plot(xx,1.96*ones(size(xx)),'--r')
 hold on
  plot(xx,-1.96*ones(size(xx)),'--r')
 ylim([min(e_star)+-1 max(e_star)+1])
 title(['Diagnostic Plot '])
```

[5.3] NON-LINEAR REGRESSION

INTRINSICALLY LINEAR

See functions FitIntrinsicallyLinear1-4

POLYNOMIAL REGRESSION

```
x; y;
n = length(x);
figure(1)
scatter(x,y)

k = 5; %poly degree
b = polyfit(x,y,5);
yfit = polyval(b,x);
e = y - yfit;
SSresid = sum(e.^2);
SStotal = (n-1)*var(y);
Rsq = 1 - SSresid/SStotal;
adjRsq = 1 - SSresid/SStotal * (n-1)/(n-k-1);
```

```
figure(1)
hold on
plot(x,yfit,'color',rand(1,3));
```

```
X = [ones(n,1) x x^2 x^3]
b = X\Y;
```

[5.4] LINEAR MULTIPLE REGRESSION

Full Model, p independent variables

```
y = data(:,1);
X = data(:,2:end);
[n p] = size(data);
XX = [ones(n,1) X];
[b,bCI,residuals,residualsInt,stats] = regress(y,XX);
yfit = XX*b;
e = y - yfit;
SSresid = sum(e.^2);
SStotal = (n-1) * var(y);
Rsq = 1 - SSresid/SStotal;
adjRsq = 1 - SSresid/SStotal * (n-1)/(n-length(b)-1);
```

Reduced Model

```
y = data(:,1);
X = data(:,2:end);
[b,se,pval,inmodel,stats,nextstep,hist]=stepwisefit(X,y);
b0 = stats.intercept;
for i=1:length(b)
     if inmodel(i) == 0
          b(i) = 0;
     end
end
b = [b0; b];
yfit = [ones(n,1) X]*b;
e = y - fit;
SSresid = sum(e.^2);
SStotal = (n-1) * var(y);
Rsq= 1 - SSresid/SStotal;
adjRsq = 1 - SSresid/SStotal * (n-1)/(n-length(b)-1);
```

```
mdl1 = LinearModel.stepwise(X,y)
mdl2 = LinearModel.stepwise(X,y,'interactions')
mdl3 = LinearModel.stepwise(X,y,'linear')
```

CHAPTER 6

[6.1] DIMENSION REDUCTION

PCA/SVD

```
XX;
[n p] = size(XX);
Y = XX - repmat(sum(XX)/n,n,1);
%Y = (XX - mean(XX))./std(XX);
[EVECTOR,EVALUE]=eig(cov(Y));
EVALUE = diag(EVALUE);
%EVECTOR = flipud(EVECTOR);
EVALUE = flipud(EVALUE);
EVECTOR = EVECTOR(:,p:-1:1);

%% Scree plot
figure(3)
plot(EVALUE,'--o')
title('Scree plot')
xlabel('index')
ylabel('eigenvalue')

%% Generate PCA component space (PCA scores)
score = (EVECTOR*Y')';
plot3(score(:,1),score(:,2),score(:,3),'.k')
title('Principal Component Scores')
xlabel('PC1')
ylabel('PC2')
zlabel('PC3')

%%  Plot first 2 PC
```

```
score2 = (EVECTOR(1:2,:)*Y')';
plot(score2(:,1),score2(:,2),'.k')
title('Principal Componenent Scores')
xlabel('PC1')
ylabel('PC2')
```

```
[n p] = size(X); [coeff,score,~,~,explained,mu] =
pca(zscore(X));
sum_explained = 0;
idx = 0;
while sum_explained < 95
    idx = idx + 1;
    sum_explained = sum_explained + explained(idx);
end

display(['The first components that explain more than 95%
of all variability: d=',num2str(idx)]);
```

ICA

```
X2 = prewhiten(X);
mdl2 = rica(X2,p,'Lamda',0.5); % ICA
%W2 = mdl2.TransformWeights;
%S2 = X2*W2;
S2 = transform(mdl2,X2) %reconstacted signal
```

[6.2] REGRESSION METHODS

```
y = data(:,1);
X = data(:,2:end);
meanY = mean(y);
meanX = mean(X);
[n, p] = size(X);
```

Ridge Regression

```
k = 0:1e-3:5e-1;
bRRv = ridge(y,X,k,0);


figure(2)
clf
plot(k,bRRv(2:end,:),'LineWidth',2)
ylim([-10 10])
grid on
xlabel('Ridge Parameter')
ylabel('Standardized Coefficient')
title('Ridge Trace')


idx = -1;
bRR = bRRv(:,1);
yfitRR = [ones(n,1) X]*bRR;
for i=1:length(k)
    yfittemp = [ones(n,1) X]*bRRv(:,i);
    if(sum(sqrt(y-yfittemp)) < sum(sqrt(y-yfitRR)))
        bRR = bRRv(:,i);
        yfitRR = yfittemp;
        idx = i;
    end
end

bRR; yfitRR;
```

LASSO

```
[bLASSOv FitInfo] = lasso(X,y,'CV',10);
idxLambdaMinMSE = FitInfo.IndexMinMSE;

%figure(3)
lassoPlot(bLASSOv,FitInfo,'PlotType','CV');
legend('show')

bLassoIntercept = FitInfo.Intercept(idxLambdaMinMSE);
bLASSO = bLASSOv(:,idxLambdaMinMSE);
fprintf('LASSO: b=[')
fprintf(' %.2f %.2f',bLassoIntercept,bLASSO)
fprintf(']\n')

yfitLASSO = bLassoIntercept + X*bLASSO;

yfitLASSO; bLassoIntercept, bLASSO
```

PLS

```
[~,~,~,~,bPLS,PCTVAR,MSE,STATS] = plsregress(X,y,p);

fprintf('PLS: b=[')
fprintf(' %.2f ',bPLS)
fprintf(']\n')

figure(4)
plot(1:p,cumsum(100*PCTVAR(2,:)),'-bo');
xlabel('Number of PLS components');
ylabel('Percent Variance Explained in y');
title('PLS')
```

```
figure(5)
yfitPLS = [ones(n,1) X]*bPLS;
residuals = y - yfitPLS;
stem(residuals)
xlabel('Observation');
ylabel('Residual');
title('PLS')

yfitPLS; bPLS;
```

OLS

```
[U S V] = svd(X-repmat(sum(X)/n,n,1),'econ');
bOLS = V*(S\U'*(y-meanY));

fprintf('OLS to detrended data: b=[')
fprintf(' %.2f ',bOLS)
fprintf(']\n')

yfitOLS = X*bOLS-meanY;

yfitOLS; bOLS;
```

PCR

```
[PCALoadings,PCAScores,~,~,explained,~] =
pca(X,'Economy',false);


sum_explained = 0;

idx = 0;

while sum_explained < 95

    idx = idx + 1;

    sum_explained = sum_explained + explained(idx);

end

fprintf('PCR: Dim. reduction to %.0f\n',idx)


bPCR = regress(y-mean(y), PCAScores(:,1:idx));

bPCR = PCALoadings(:,1:idx)*bPCR;

bPCR = [mean(y) - mean(X)*bPCR; bPCR];

yfitPCR = [ones(n,1) X]*bPCR;


yfitPCR; bPCR;
```

APPENDIX – MATLAB HELP

- `mink(A,k)` – finds k minimum elements of vector A
- `maxk(A,k)` – finds k maximum elements of vector A
- `plot(x,y,'color',rand(1,3));`
- ```
  Legend = cell(5,1);
  Legend{1} = strcat('This is no ',num2str(1));
  legend(Legend);
  ```
- `text(xdim,ydim,'Add text here')` – adds text to plot
- `rnd = unidrnd(n,m,1)` – creates m numbers from 1 to n
  `x=x(rnd)`
- `line([-6 -2],[3 3])` – draws a horizontal line
- `yline(5)` – draw a horizontal line
- `x=1:12; plot(x,5*ones(size(x)))` – draws a horizontal line
- `plot([5 5],[0 10],'g')` – draws a vertival line
- ```
  v=0:1000;
  [x1 x2]=meshgrid(v,v);
  z=x1*x2;
  surf(x1,x2,z); - draws 3d
  ```
- `length(find(H==1))` – finds number of 1 in H
- `find(H==0)` – returns a vector with 1 in position i if H(i)=0, and with 0 in position i otherwise
- ```
  tittxt = str2mat('Add text here');
  fprintf('Text is %s\n',deblank(tittxt(:,1)));
  ```
- ```
  figure(1)
  suptitle('Title of all')
  subplot(2,2,1) %Creates 2x2 grid for plots
  plot(…)
  title(…)
  subplot(2,2,2)
  plot(…)
  title(…)
  ```

- Line Specification

'-'	Solid line (default)
'--'	Dashed line
':'	Dotted line
'-.'	Dash-dot line
'+'	Plus sign
'o'	Circle
'*'	Asterisk
'.'	Point
'x'	Cross
'square' or 's'	Square
'diamond' or 'd'	Diamond
'^'	Upward-pointing triangle
'v'	Downward-pointing triangle
'>'	Right-pointing triangle
'<'	Left-pointing triangle
'pentagram' or 'p'	Five-pointed star (pentagram)
'hexagram' or 'h'	Six-pointed star (hexagram)
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White