| Function; Variable; Reserved Keywords done by: Ling 1 Dec 19 | | | |
| --- | --- | --- | --- |
|  | **SAS** | **R** | **SPSS** |
| Filepath | Filepath = F:\folder\file\ | F:/folder/file |  |
| Missing entry | . | NA | . |
| Comment lines | /\* comments \*/ | # comments | - |
| Importing data | data varname;  infile 'filepath' delimiter="," firstobs=2 obs=n; [obs is the final line] | var1 = read.table('filepath', header=TRUE, sep=",", col.names, row.names)  [attach column names] attach(var2) | "File" -> "Open" |
| Header names | input col1 col2$ @@;  run; | [col names] names(df1) = c('name1','name2')  [row names]  row.names(df1) = c('name1','name2') | \*edit directly |
| Manual entry | datalines;  data1 data2 data3  ; | [create vectors]  var1 = c(data1,data2,data3);  var4 = seq(from=n, to=m, by=x)  [create numeric] var2 = numeric(n)  [replicates elements] var3 = rep(list,times)  [convert vectors to matrix]  dim(vector) = c(row,col)  [create matrix]  var5 = matrix(vector,nrow=n,ncol=m)  [create data frames]  var6 = data.frame(vector/matrix)  [manual entry] var7 = scan() | "Variable View" -> Define variables  "Data View" -> Enter data |
| Fixed format entry | input col1 1-2 col2$ 4; | var1 = read.fwf('filepath',width=c(1,2,3)) | "File" -> "Open" -> "Fixed width" |
| Creating subset | set varname; (filepath if SAS dataset)  if condition1 then output; | \*reassign variable | "Data" -> "Merge Files" |
| Filter | if condition1 then output;  \*note: var must be created first | var[condition,] | [Data View]  "Data" -> "Select Cases" -> "If condition is satisfied" -> "If" -> enter conditions |
| Create new col | col\_name = (a+b)/2; | [expand dataframe] var $col = var  [bind a row to matrix] var3 = rbind(var1,var2)  [bind a col to matrix] var3 = cbind(var1,var2) | "Edit" -> "Insert Variable" |
| Keeping subset | keep col1 col2 col3; | var1[rows,cols] \*could be logics(conditions) too  {note: when filter, apply on rows} | - |
| Dropping subset | drop col1 col2 col3; | "Edit" -> "Clear" |
| Concatenating data set | set var1 var2; | [combining data frames by rows]  var3 = rbind(var1,var2) | [Data View] "Data" -> "Merge Files" -> "Add Cases" |
| If-else | if condition then outcome;  else if conditon then output; | if(condition) {statement}  else if (condition) {statement}  else {statement} | \*applicable in filter |
| Loop | do var = start to end;  *statements*;  end;  e.g.  do batch = 1 to n;  do treat = 1 to n;  input value @@;  output;  end;  end; | [while loop] while(*condition*){*statements*}  [for loop] for (var in 1:n){*statement*} | - |
| Merging data | [Must be sorted by criteria first]  merge var1 var2;  by criteria; | [merge data by extending cols]  var6 = merge(var4,var5,by='criteria',all=TRUE) | [sorted data] "Data View" -> "Data" -> "Merge Files" -> "Add Variables" |
| Output data | \*change the dataname to a permanent address | [save output on screen]  sink('filepath') *operations* sink()  [print output] cat(value1,value2,value3,"\n")  [save dataframes] write.table(var1,'filepath',quote=FALSE) | "File" -> "Save" |
| Sorting | proc sort data=name;  by descending col1; | var1[order(criteria),]  var2[rev(order(criteria)),]  rank(var) | "Data" -> "Sort Cases..." |
| Writing functions | - | functionname = function(input1,input2){  *statement*  return(output)} | - |
| Transformation | \*refer to create new col | \*create new var  \*e.g. BMI = height/weigt\*\*2 | [Calculation] "Transform" -> "Compute Variable"  [Recode] "Transform" -> "Recode into Different Variable"  {note: Click "Output variable are strings" if output values are characters} |
| Format values | proc format; \*gives lables to values  value $value\_name '1' = 'Name1';  value value\_name 1-20 = "range"  . = "No reply"  Other = "Out of range";  data var;  label col\_name = 'col name'  col\_name = 'col name'; #label headers  format col\_name $ value\_name.  col\_name $ value\_name.; | #value labels  label = c("label1","label2")  var.group = label[var] #making use of indexing  #recode var  value = ifelse(test=,yes=,no=) #convert value to index |  |
| Logical Operators | |  |  |  |  | | --- | --- | --- | --- | | equal | = | greater than and equal | >= | | less than | < | not equal | ^= | | less than and equal | <= | negation(NOT) | ^ | | greater than | > |  |  | | |  |  |  |  | | --- | --- | --- | --- | | equal | == | not equal | != | | less than | < | negation(NOT) | ! | | less than and equal | <= | AND | & | | greater than | > | OR | | | | greater than and equal | >= |  |  | | - |
| Modify value |  | var[row,col] = value |  |
| Applying functions |  | list apply, apply function to a list, recoding the list  lapply(list, function)  table apply  tapply(vector, index, function)  \*the index works like grouping variable  apply(array, margin, function)  \* ﻿margin: 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns. Where X has named dimnames, it can be a character vector selecting dimension names. |  |

| Function; Variable; Reserved Keywords | | | SAS | SPSS |
| --- | --- | --- | --- | --- |
| Describing numerical data | | | | |
| Numerical descriptive measures | - Location  - Variability  - Other measures | | proc means data=var n mean std min max stderr maxdec=n;  title "this is a title";  var col\_name;  (class col\_name; \*different groups)  OR  proc univariate data=var;  title "this is a title";  var col\_name; | "Analyze" -> "Descriptive Statistics" -> "Frequencies" |
| Graphical methods | - Histogram | | proc univariate data=var;  var col\_name;  histogram col\_name / midpoints=n to m by 10 normal;  class col\_name; \*seperate by group  qqplot col\_name; | "Analyze" -> "Descriptive Statistics" -> "Explore..."  OR ~->"QQ Plots..."  "Graphs" -> "Legacy Dialogs" -> "Histogram"/"Box plot"  \*add <Factor List> for 2 variable analysis |
| - Boxplot, Stem and leaf plot, QQ(quantile-quantile) plot | | proc univariate data=var plot;  var col\_name;  OR  proc boxplot data=var;  plot col\_name\*col\_name; |
| - Scatter plot for bivariate data | | proc sgplot data=var;  title "Title";  by col\_name; #different graph  scatter x=col\_name y=col\_name / group=criteria; #same graph | "Graphs" -> "Chart Builder"  \*need to drag x and y factors, set colour  OR in "Legacy Dialogs" |
| Robust Location estimator | - Trimmed mean | | proc univariate data=var trimmed=0.2 winsorized=0.2;  var col\_name; | - |
| - Winsorized mean | | - |
| - Huber's M-estimators | | - | "Analyze:" -> "Descriptive Statistics" -> "Explore..." |
| - Tukey's bisquare estimator | | - |
| - Hampel's M-estimator | | - |
| Robust measures of scale parameter | - Interquartile Range (IQR) | | proc univariate data=var robustscale;  var col\_name; |  |
| - Median Absolute Deviation (MAD) | |  |
| - Gini's mean difference | |  |
| Categorical data | - Frequency tables | | proc freq data=var;  title "Title";  tables col\_name -- col\_name; | "Analyze" -> "Descriptive Statistics" -> "Frequencies" |
| - Contingency tables (two-way frequency table) | | tables col\_name\*col\_name; | "Analyze" -> "Descriptive Statistics" -> "Crosstabs..." -> "Stat |
| - Chi-square tests | | tables col\_name\*col\_name / chisq;  weight col\_name; #for manual entry summarised table | "Data" -> "Weight Cases" -> contingency |
| - Charts | |  |  |
| Categorical data (paired data) | - McNemar’s test | | proc freq data=var;  tables before\*after/agree;  weight count; #for frequency counts | “Analyze” -> “Descriptive Statistic” -> “Crosstabs” |
| - Bar chart | | proc sgplot data=var;  vbar col\_name;  #for two way frequency table  panelby col\_name;  vbar col\_name /group=group\_name; | “Graphs” -> “Legacy Dialogs..” -> “Bar…” |
| Numerical data  Test for location | Parametric test  - One sample t-test (normal population) | | proc univariate data=var mu0=value;  var col\_name; | "Analyze" -> "Compare Means" -> "One Sample t tests..." |
| Nonparametric test  - Sign test | | "Analyze" -> "Nonparametric Tests" -> "Legacy Dialogs" -> "2 Related Samples"  \*note: must duplicate sample mean in a new col first |
| - Sign rank test | |
| Tests comparing two groups | Two independent sample  - Parametric Test (t-test) | | proc ttest data=var hu0=value sides=2/L/U;  class col\_name; #two groups  var col\_name; #values | "Analyze" -> "Compare Means" -> Independent Sample T tests..." |
| - Nonparametric tests  (Wilcoxon rank sum test)  (Mann-Whitney U-test) | | proc npar1way data=var wilcoxon;  class col\_name; #two groups  var col\_name; #values  exact wilcoxon; | #recode into numeric variable  "Transform" -> "Recode" -> "Into different variable..."  #wilcoxon rank sum test  "Analyze" -> "Nonparametric Tests" -> "Legacy Dialogs" -> "2 Independent Samples..." |
| Related samples  - Paired t-test | | proc ttest data=var mu0=value sides=2/L/U;  paired var1\*var2;  #or create a difference and run 1 var t-test | "Analyze" -> "Compare Means" -> "paired Sample T test..." |
| Related samples: Nonparametric tests  - Wilcoxon signed rank test | | #create a difference at data import stage  proc univariate data=var;  var col\_name; | "Analyze" -> "Nonparametric Tests" -> "Legacy Dialogs" -> "2 Related Samples..." |
| Tests comparing three groups or more | Parametric Test  - ANOVA table | | proc glm data=var;  class col\_name;  model y=x; | “Analyze” -> “Compare Means” -> “One-way ANOVA...”  “Options”, “Descriptive” |
| - LSD | | means col\_name/ lsd; | “Analyze” -> “Compare Means” -> “One-way ANOVA...”  “Post Hoc..”, “LSD” |
| - Contrast | | contrast “X vs. Y and Z” y\_col -2 1 1;  contrast “Y vs. Z” y\_col 0 1 -1; | “Analyze” -> “Compare Means” -> “One-way ANOVA...”  “Contrast” |
| Testing for Equal Variances  - Levene’s test & Bartlett test | | means col\_name/ hovtest = levene hovtest = bartlett; | “Analyze” -> “Compare Means” -> “One-way ANOVA...”  “Options”, “Homogeneity of variance test” |
| Testing for Independence and Normality Assumptions | | \*obtain the residuals;  proc glm data=var;  class col\_name;  model y=x;  output out=var2 p=yhat r=resid;  \*test for normality;  proc univariate data=var;  var resid;  histogrm resid/ normal noplot;  qqplot resid;  \*residual plots;  proc sgplot data=var;  scatter x=yhat y=resid; | “Analyze” ->”General Linear Model” -> “Univariate..”  “Dependent Variable”, “Fixed Factor” |
| Nonparametric 1-way test  - Kruskal-Wallis Test | | proc npar1way data=var;  class col\_name;  var col\_name;  exact wilcoxon; | “Analyze” -> “Nonparametric Tests” -> “Legacy Dialogs” -> “K Independent Samples...”  “Test Variable List”, “Grouping Variable”  “Define Range” |
| Regression Analysis | Pearson Correlation Coefficient | | proc corr data=var nosimple;  var col\_name1 col\_name2 col\_name3; | “Analyze” -> “Correlate” -> “Bivariate...”  “Variables window”, “Pearson” |
| Simple regression | | proc reg data=var;  model y = x;  output out=var p=yhat r=resid;  quit; | “Analyze” \_> “Regression” -> “Linear” |
| Testing assumptions  - Linear relationship: scatter plot | | proc sgplot data=var;  scatter y=col\_name x=col\_name; | “Graphs” -> “Legacy Dialogs...” -> “Scatter/plot”  “Simple Scagter” -> “Define” |
| - Normality and independence | | proc sgplot data=var;  scatter y=resid x=yhat;  proc sgplot data=var;  scatter y=resid x=height; | “Analyze” -> “Regression” -> “Linear”  [and save residuals]  “Analyze” -> “Descriptive Statistics” -> “QQ plots...” |
| - Prediction | Draw fitted lines | proc sgplot data=var;  pbspline y=col\_name x=col\_name;  \*pbspline = penalized beta spline method; |  |
| Case1: value in dataset | Add statement output out=var p=yhat r=resid and print dataset to search | “Analyze” -> “Regresion” -> “Linear”  “Save”, “Unstandardized” |
| Case2: value not in | Add data point into dataset with the missing values (using . for missing value) | Add missing data point |
| - Transform | Quadratic | data var2;  set var1;  x2 = x\*\*2;  proc reg data=var;  model y=x x2; | “Transform” -> “Compute Variable...” |
| Log | data var2;  set var1;  lx = log(x);  proc reg data=var;  model y=x lx; |

| Function; Variable; Reserved Keywords | | | R | |
| --- | --- | --- | --- | --- |
| Describing numerical data | | | | |
| Numerical descriptive measures | | | | |
| - Location | - Mean | | mean(var) | summary(var)  Min/1stQ/Median/Mean/3rdQ/Max |
| - Median | | median(var) |
| - Mode | | var[order(var)[1:n]]#smallest 5 | |
| - Variability | - Variance | | var(var) | |
| - Standard deviation | | sd(var) | |
| - Range | | range(var) | |
| - Interquartile range | | IQR(var) | |
| - Coefficient of Variation | | cv = function(x) sd(x)/mean(x)\*100 | |
| - Other measures | - Min and Max | | min(var); max(var) | |
| - First & third quartile | | quantile(var, level) | |
| - Percentiles | |
| n = length(x); mn = mean((x-mean(x)^n)) | | | |
| - Skewness [measure the symmetric] | | skew = function(x) {  sk = m3/m2^(3/2)\*sqrt(n\*(n-1))/(n-2)  return(sk)} | |
| - Kurtosis [measure the tail] | | kurt = function(x) {  ku = (n-1)/((n-2)\*(n-3))\*((n+1)\*m4/m2^2-3\*(n-1))  return(ku)} | |
| Graphical methods | | | | |
| - Histogram | par(mfrow=c(2,1)) #2 graphs in one column in 1 page  hist(var, include.lowest=TRUE, freq=TRUE, col='color', main=paste("Title"), sub=paste("sub-title"), xlab ="x", ylab ="y", axes=TRUE)  *#normal curve imposed on the histogram*  xpt = seq(n,m,0.1); ypt=dnorm(seq(n,m,0.1),mean(col\_name),sd(col\_name))  aypt=ypt\*length(col\_name)\*10 *#scaling the line to match area of histogram, 10 is the width of bin*  lines(xpt,aypt)  par(mfrow=c(1,1)) #1 graphs in one column in 1 page | | | |
| - QQplot | qqnorm(col\_name); qqline(col\_name) | | | |
| - Boxplot | boxplot(col\_name~col\_name2) #~means split by group | | | |
| - Stem and leaf plot | stem(col\_name) | | | |
| - Scatter plot for bivariate data | plot(x=col\_name,y=col\_name, xlim=c(lower,upper), ylim=c(lower,upper), axes=FALSE, pch=0, col=2)  par(new=TRUE) #plot new graph and old graph together | | | |
| Robust Location estimator | | | | |
| - Trimmed mean | | | mean(var, trim=0.2) | |
| - Winsorized mean | | |  | |
| - Huber's M-estimators | | |  | |
| - Tukey's bisquare estimator | | |  | |
| - Hampel's M-estimator | | |  | |
| Robust measures of scale parameter | | | | |
| - Interquartile Range (IQR) | | | IQR(var)/1.34898 | |
| - Median Absolute Deviation (MAD) | | | median(abs(x-median(x)))  mad(x) | |
| - Gini's mean difference | | |  | |
| Categorical data | | | | |
| - Frequency tables | | | table(var) | |
| - Contingency tables | | | table(var,var2) | |
| - Fisher test | | | fisher.test(table(var),  alternative="less/greater/two.sided") | |
| - Chi-square tests | | | chisq.test(table(var), correct=F) #applied w/o continuity correction | |
| - Charts | | |  | |
| Paired data | | - McNemar’s test | mcnemar.test(table1, correct=T) | |
| - Bar chart | barplot(height= c(table1), names.arg=c(“Nam1”,”Name2”))  #two-way frequency table  barplot(tablecount, beside=T, col=c(1,8))  legend(“top-left”, c(“Name1”,”Name2”), fill=c(1,8)) | |
| Numerical data | | | | |
| Tests for location | | Parametric test  - One sample t-test | t.test(var, mu=value, alterntive="two.sided/less/greater") | |
| Nonparametric test  - Sign test | testvalue = var[var!=mu0]  binom.test(sum(testvalue>mu0),  length(testvalue)) | |
| - Signed Rank test | wilcox.test(testvalue, mu=value, alternative="two.sided") | |
| Two independent samples | | - t-test | #first create different groups  #test for variances  var.test(var1, var2)  #t-test  t.test(var1,var2,mu=0,var.equal=T/F,  alternative="two.sided") | |
| - Nonparametric (Wilcoxon rank sum test) | #first create different groups  wilcox.test(groupa, groupb)  or wilcox.test(y~x) | |
| Two related samples | | - Paired t-test | t.test(var1,var2,mu=0,paired=T) | |
| Nonparametric test  - Signed test | #create a diff first  diff = varA-varB  ncount = sum(diff>0)  binom.test(ncount, length(diff), 0.5) | |
| - Signed rank test | wilcox.test(diff) | |
| Three or more independent samples | | Parametric test  - ANOVA | model1 = anov(y~x)  summary(model1)  tapply(x,y,mean); mean(x); tapply(x,y,mean)-mean(x); boxplot(y~x) | |
| - LSD | group.1 = y[x==”1”]; group.2 = y[x==”1”]; group.3 = y[x==”3”];  group.mean = tapply(x,y,mean)  treat.group = cbind(group.1, group.2, group.3)  #d=(N-k) degree of freedom, where k is the no. of groups  mse = sum(model1$res^2)/d  lsd = qt(0.975,d)\*sqrt(mse\* (1/n1 + 1/n2) )  diff = mean(treat.group[,i]) - mean(treat.group[,j])  if(abs(diff)>lsd) {cat(“There is significant difference between groups”)} | |
| - Contrast | contrasts(x) = matrix(c(2,-1,-1,0,  -1,1),nrow=3)  modelc = lm(y~x)  summary(modelc) | |
| Testing for Equal Variances  - Levene’s test | group.means = tapply(y,x,mean)  gmean = group.means[x]  #Absolute deviation from group meams is used  words.abs.dev = abs(y-gmean)  modellev = aov(words.abs.dev~x)  summary(modellev) | |
| - Bartlett’s Test | bartlett.test(y,x) | |
| Test for independence and Normality assumptions | #Test for normaity (Kolmogorove-Smirnov test)  resid=model1$res  ks.test(resid,”pnorm”,mean(resid),sd(resid))  #Model checking: residual plots  fv = model1$fitted  par(mfrow=c(2,1))  plot(fv,resid); abline(h=0,lty=2)  qqnorm(resid); qqline(resid,lty=2)  par(mfrow=c(1,1)) | |
| Nonparametric 1-way Test  - Kruskal-Wallis Test | kruskal.test(y,x)  or  kruskal.test(y~x) | |
| Simulation Study | | | | |
| Conditon of interest  - Compare 3 estimators for location through a simulation study  - 3 estimators are: sample mean, sample median, sample 10% trimmed mean  - Underlying condition: N(1,1)  - Sample size: 15  - Simulation size: 1000 | | | | |
| Comparing the estimators | | Environment set up | ns = 1000; n = 15; mu = 1; sd = 1;  meax = numeric(ns); medx = numeric(ns);  trmx = numeric(ns); stdx = numeric(ns);  set.seed(12345) | |
| Compute the numeric value of the estimators | for (i in 1:ns){  x = rnorm(n, mu, sd)  meax[i] = mean(x)  medx[i] = median(x)  trmx[i] = mean(x, trim=0.1)  stdx[i] = sd(x) } | |
| Compute the mean of estimators | simumean = apply(cbind(meax,medx,trmx),2,mean) | |
| Compute the sd of estimators | simustd = apply(cbind(meax, medx, trmx), 2, sd) | |
| Compute bias | simubias = simumean – rep(mu,3) | |
| Compute MSE | simumse = simubias^2 + simustd^2 | |
| Presentation | sumdata = rbind(c(mu,mu,mu), ns, simumean, simustd, simubias, simumse)  col.name = c(“mean”,”median”,”10% tmean”)  row.name = c(“True value”, “No. of simu”, “MC Mean”, “MC sd”, “MC Bias”, “MC MSE”)  dimnames(sumdata) = list(row.name,col.name)  round(sumdata,4) | |
| Checking coverage probability of confidence interval | | getting | t05 = qt(0.975,n-1) | |
| Compute CI | lower = meax-t05\*stdx/sqrt(n)  upper = meax+t05\*stdx /sqrt(n) | |
| Test logic | coverage = sum(lower <= mu & upper >= mu) | |
| Compute coverage | coverage = coverage/ns | |
| Checking size of t-test | | Calculate t stat | ttests = (meanx-mu)/(stdx/sqrt(n)) | |
| Compute rejection | reject = abs(ttests) > t05 | |
| Calculate Size | size = sum(reject)/ns | |
| Checking power of t-test when | | Generate data under | ns = 1000; n = 15; mu0 = 1; sigma = 1;  mu1 = mu0 + 0.5\*sigma  meanx = numeric(ns); stdx = numeric(ns);  set.seed(12345)  for (i in 1:ns){  x = rnorm(n, mu1, sigma)  meanx[i] = mean(x)  stdx[i] = sd(x) } | |
| Calculate t stat | t05 = qt(0.975,n-1)  ttests = (meanx-mu0)/(stdx/sqrt(n)) | |
| Compute rejection | #alternate method of calculating rejection  reject = ttests >= t05 | ttests <= -t05 | |
| Calculate power | power = sum(reject)/ns | |
| Bootstrap method | | | | |
| Generate bootstrap samples | | Compute true value | options(digit=4)  theta.hat = corr(var1,var2) | |
| Generate bootstrap samples  & compute stat of interest  (corr in this case) | B = 1000; n = nrow(data);  theta.b = numeric(B);  for (b in 1:B){  index = sample(1:n, size=n, replace=TRUE)  bsample1 = var1[index]  bsample2 = var2[index]  theta.b[b] = corr(bsample1, bsample2) | |
| Bootstrap Estimation of Standard Error | | Compute sd | sd(theta.b) | |
| Bootstrap Estimation of Bias | | Compute bias | bias = mean(theta.b) – theta.hat | |
| Alternate method for SE and Bias | | Using Library boot | bcor = function (data, b) {  return(cor(data[b,1],data[b,2])) }  library(boot)  boot.cor = boot(data, statistic=bcor, R=5000)  #gives original value from sample, bias and standard error | |
| Basic bootstrap Confidence interval | | Compute lower and upper quantile | alfa = 0.05  low = quantile(theta.b,alfa/2)  high = quantile(theta.b,1-alfa/2) | |
| Compute CI | lower = 2\*theta.hat-high  upper = 2\*theta.hat-low | |
| Percentile bootstrap confidence interval | | Compute lower and upper quantile | alfa = 0.05  low = quantile(theta.b,alfa/2)  high = quantile(theta.b,1-alfa/2) | |
| Alternate method for CI | | Using Library boot | bcor = function (data, b) {  return(cor(data[b,1],data[b,2])) }  library(boot)  boot.cor = boot(data, statistic=bcor, R=5000)  boot.ci(boot.cor,type=c(“basic”,”norm”,”perc”))  #gives normal, basic and percentile CI  #must generate boot.cor object first | |
| Numerical methods in R | | | | |
| Finding Root | | One-dimension  \*uniroot only finds 1 root in region  \*function must have opposite signs in two end points | a = 0.5; n = 20;  fy = function(y) {a^2+y^2+2\*a\*y/(n-1)-(n-2)}  rt = uniroot(fy,lower=0,upper=100)  root = rt$root #to get root  value = rt$f.root #to get the value of fn | |
| Finding roots  \*polyroot finds multiple roots, including complex roots | #input the coefficient of polynomial  a0 = a^2 – (n-2); a1 = 2\*a/(n-1); a2 = 1;  polyroot(c(a0,a1,a2))  # is the coefficient of | |
| Integration | | One variable integral | fcn = function(y,mu,sigma) {  abs(y)\*exp(-(y-mu)^2/(2\*sigma^2)) / sqrt(2\*pi)/sigma }  integrate(fcn,lower=-Inf, upper=Inf, mu=0, sigma=1) | |
| Optimiation | | One-dimentional | f = function(x) {log(x+log(x))/log(1+x)}  optimize(f,lower=4,upper=8,maximum=TRUE) | |
| Two-dimensional  \*”optim” performs minimization by default | LL = function(theta, sx, slogx, n) {  r = theta[1]; lambda = theta[2]  loglik = n\*r\*log(lambda)+(r-1)\*slogx- lambda\*sx-n\*log(gamma(r))  return(loglik)}  #or return(-loglik) if lazy to use fnscale  n = 200; r = 5; lambda = 2;  theta.start = c(1,1)  x = rgamma(n,shape=r,rate=lambda)  out = optim(par=theta.start,fn=LL,sx=sum(x), slogx=sum(log(x)),  n=n,control=list(fnscale=-1))  r = out$par[1]; lambda = out$par[2]  result = out$value | |
| Regression Analysis | | | | |
| Pearson Correlation Coefficient | | | cor(dataframe, method=”person”)  #test if true correlation is 0  cor.test(var1,var2) | |
| Simple regression | | | model1 = lm(y~x)  #to see sum of squared regression and error  anova(model1)  #to see R-squared and linear model, F-test  summary(model1) | |
| Testing assumptions  - Linear relationship: scatter plot | | | model1 = lm(y~x)  par(mfrow=c(2,2))  #scatter plot  plot(y~x); abline(model1);  title(“Scatter plot and Regression Line”) | |
| - Normality and independence  observe if there’s any trend in residuals, it shouldn’t have | | | #residual plot  rs = model1$resid; fv = model1$fitted;  plot(rs~x);abline(h=0)  #normal QQ plot  qqnorm(rs); qqline(rs)  par(mfrow=c(1,1)) | |
| - Prediction | | case1: data in set | model1$coeff[1]+value\*model1$coeff[2] | |
| case2: data not in | predict(model1,data.frame(x=value))  #or the previous method | |
| - Transformation | | Investigate rs | plot(x,y);lines(smooth.spline(x,y)) | |
|  | | quadratic | x2 = x^2;  model2 = lm(y~x+x2)  anova(model2);summary(model2) | |
| log | lx = log(x);  model3 = lm(y~lx)  anova(model3);summary(model3) | |

| Function; Variable; Reserved Keywords done by: Ling 1 Dec 19 | | | |
| --- | --- | --- | --- |
|  | | **SAS** | **R** |
| Simulation | | | |
| Generate random numbers | Linear congruential generators | data try1;  seed = 1234;  do i = 1 to 10;  x = ranuni(seed);  output;  end; keep x; | ran = NULL  for (i in 1:n){  x1 = (a\*x0+c)%%m  x0 = x1  ran = c(ran,x1/m)} |
| Uniform random numbers | data unif;  seed = 1234; call streaminit(seed);  n=100; a=0; b=10;  do i = 1 to n;  x = a + (b-a) \* rand(“UNIFORM”); output;  end; keep x; | #uniform with (0,1)  x = runif(1000)  #uniform with (a,b)  set.seed(1234)  x = runif(n,a,b) |
| Normal( | data norm;  seed = 1234; call streaminit(seed);  n=100; mu=0; sigma=1;  do i = 1 to n;  x = rand(“NORMAL”, mu, sigma); output;  end; keep x; | x = rnorm(n,mean=mu,sd=sigma) |
| Exponential( | data expno;  seed = 1234; call streaminit(seed);  n=100; lambda=5;  do i = 1 to n;  x = lambda \* rand(‘EXPONENTIAL’); output;  end; keep x; | x = rexp(n,mean=lambda) |
| Gamma( | data gammano;  seed = 1234; call streaminit(seed);  n=100; alpha=1; beta=2;  do i = 1 to n;  x = beta \* rand(‘GAMMA’, alpha); output;  end; keep x; | x = rgamma(n,shape=alpha,scale=beta) |
| Chi-square(p) | data chisqno;  seed = 1234; call streaminit(seed);  n=100; df=10;  do i = 1 to n;  x = rand(‘CHISQUARE’, df); output;  end; keep x; | x = rchisq(n,df=p) |
| Beta( | data betano;  seed = 1234; call streaminit(seed);  n=100; alpha=2; beta=3;  do i = 1 to n;  x = rand(‘BETA’, alpha, beta); output;  end; keep x; | x = rbeta(n,shape1=a,shape2=b) |
| t(k) | data tno;  seed = 1234; call streaminit(seed);  n=100; df=5;  do i = 1 to n;  x = rand(‘T', df); output;  end; keep x; | x = rt(n, df=k) |
| F(m,n) | data fno;  seed = 1234; call streaminit(seed);  n=100; df1=5; df2=10;  do i = 1 to n;  x = rand(‘F', df1, df2); output;  end; keep x; | x = rf(n, df1 = n1, df2 = n2) |
| Binomial(n,p) | data binomno;  seed = 1234; call streaminit(seed);  ns=100; n=10; p=0.3;  do i = 1 to ns;  x = rand(‘BINOMIAL’, p, n); output;  end; keep x; | x = rbinom(n, size, prob = p) |
| Poisson( | data poisno;  seed = 1234; call streaminit(seed);  n=100; lambda=3;  do i = 1 to n;  x = rand(‘POISSON’, lambda); output;  end; keep x; | x = rpois(n, lamba) |
| Hypergeo(n,N,S) | data hypergeono;  seed = 1234; call streaminit(seed);  ns=100; popnsize=3; numbsucc=5; samplesize=3;  do i = 1 to ns;  x = rand(‘HYPERGEOMETRIC’, popnsize, numbsucc, samplesize);  output;  end; keep x; | x = rhyper(nn, m=S, n=N, k=n) |
| NBinom(r,p) | data negbinomno;  seed = 1234; call streaminit(seed);  n=100; p=0.3; k=5;  do i = 1 to n;  x = rand(‘NEGBINOMIAL’, p, k); output;  end; keep x; | x = rnbinom(n, size=r, prob=p) |
| Simulation Study | | | |
| Conditon of interest  - Compare 3 estimators for location through a simulation study  - 3 estimators are: sample mean, sample median, sample 10% trimmed mean  - Underlying condition: N(1,1)  - Sample size: 15  - Simulation size: 1000 | | | |
| Size of t-test | Generate sample set | data simu;  seed = 123; ns=1000; n=15; mu=0; sigma=1;  call streaminit(seed);  do mcrep = 1 to ns;  do i = 1 to n;  value = rand(“NORMAL”,mu,sigma); output;  end;  keep mcrep value;  end; |  |
| Calculate t test | proc sort data=simu;  by mcrep;  proc univariate data=simu noprint mu0=0;  by mcrep;  var value;  output out=outtest probt = p; |  |
| Calculate rejection | data outtest1;  set outtest;  reject = (p<0.05); |  |
| Calculate rejection rate | proc means data=outtest1 noprint;  var reject;  output out = results mean=rejrate;  proc print data=results;  var \_freq\_ rejrate; |  |

|  |  |  |  |
| --- | --- | --- | --- |
| done by: Ling 1 Dec 19 | | | |
| Describing numerical data | | | |
| Numerical descriptive measures | - Location | - Mean  - Median  - Mode | |
| - Variability | - Variance or standard deviation  - Range  - Interquartile range  - Coefficient of Variation | |
| - Other measures  unbiased estimator of kurtosis: | - Minimum and Maximum  - First quartile and third quartile  - Percentiles  - Skewness [measure the symmetric]  - Kurtosis [measure the tail] | unbiased estimator of skewness: |
| Graphical methods | - Histogram |  | |
| - Boxplot, Stem and leaf plot |  | |
| - Scatter plot for bivariate data |  | |
| Robust Location estimator | | | |
| Robust location estimators | - Trimmed mean (top-left) |  | |
| - Winsorized mean |
| - Huber's M-estimators (top-right) |
| - Tukey's bisquare estimator (bottom-left) |
| - Hampel's M-estimator (bottom-right) |
| Robust measures of scale parameter | - Interquartile Range (IQR) |  | |
| - Median Absolute Deviation (MAD)  (median of the diff in median) |  | |
| - Gini's mean difference | e.g. G = (a1 + .. + an)/n | |
| Categorical data | | | |
|  | - Frequency tables |  | |
| - Contingency tables |  | |
| - Chi-square tests (single tail test) |  | |
| - Charts |  | |
| Paired Data | Same subjects under two different conditions | | |
| - McNemar’s test |  | |
| - Bar Chart |  | |
| Numerical data | | | |
| One Sample Tests (Hypothesis testing) | | | |
| Location | Parametric test  - One sample t-test (normal population) |  | |
| Nonparametric tests  - Sign test  - Sign rank test |  | |
| Two comparing two groups (Hypothesis testing) | | | |
| Independent samples | Two-sample t-test (normal population) |  | |
| Wilcoxon rank-sum test (non-parametric) |
| Related samples | Paired t-test (normal population) |  | |
| Sign Test or Wilcoxon signed rank test(non-parametric) |
| 3 or more independent groups (One-way analysis of variance) | | | |
| Parametric method | One-way analysis of variance  - ANOVA |  | |
| Assumptions of ANOVA  (based on F-test)  & Model Checking | 1. Random sample 2. Equal variances for all the groups    1. Bartlett test or Levene’s test 3. Independence of errors    1. Residual plots       1. QQplot on residuals       2. Plot residuals against the groups 4. Normal distribution of errors    1. Normality test on residuals 5. Additivitiy of treatment effects | |
| Multiple comparison tests  - Least significant difference (LSD) | LSD for  We conclude is different from if | |
| - Contrast method | A contrast of means is a linear combination of the means such that the coefficients of the means such that the coefficients of the means should sum up to zero  e.g., | |
| Others  - Duncan’s multiple-range test  - Student-Newman-Keul’s multiple-range test  - Scheffe’s multiple-comparison procedure | | |
| Nonparametric method | Kruskal-Wallis test |  | |
| Simulation | | | |
| Usage | - Checking distribution theory | Theory:  Take a sample of size 4 from normal distribution with mean 0  Simulation:  Generate 1000 random samples with size 4. Compute statistic T and construct a historygram.  Superimpose the t(3) curve and compare | |
| - Comparing estimators | Question:  Comparing robustness of estimators in different conditions (underling distributions)  Consider  Simulation:  Generate random samples. Compute MSE for each estimators. | |
| - Buffon’s needle experiment | Question:  Compute a question of interest through a function.  Consider the probability that needle intersects a line  Simulation:  Simulate the experiement by N times and count the chances of success  n/N give the estimate of the probability of success | |
| Generate numbers | | | |
| Random number | - Congruential generators | Uniform random numbers: | |
| Uniform numbers | - Uniform random numbers |  | |
| Non-uniform random numbers | - Inversion method | If X has a continuous distribution function F(x),  where , then  Generate U from Uniform(0,1) and set | |
| - Exponential distribution |  | |
| - Weibull distribution |  | |
| - Cauchy distribution |  | |
| Generate Normal random variable | - Box-Muller Algorithm | Generate and from uniform(0,1)  Now we have X and Y, two independent standard normal variables | |
| Random variable from other random variables | - Cauchy distributions |  | |
| - Chi-square distribution |  | |
| - Student’s t-distribution |  | |
| - F distribution |  | |
| Functions | Other functions to generate the random no. | <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/Distributions.html>  http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a001466748.htm | |
| Generate random numbers | Uniform(a,b) |  | |
| Normal( |  | |
| Exponential( |  | |
| Gamma( |  | |
| Chi-square(p) |  | |
| Beta( |  | |
| t(k) |  | |
| F(m,n) |  | |
| Binomial(n,p) |  | |
| Poisson( |  | |
| Hypergeo(n,N,S) |  | |
| NBinom(r,p) |  | |
| Simulation Studies in Stats | | | |
| Rationale  (in statistics) | Properties of estimators | - Bias  - Consistent  - Sampling variance  - Comparison with competing estimators on bias, precision etc  - Does confidence interval achieve the advertised nominal level of coverage (e.g.95%)? | |
| Properties of hypothesis testing procedures | - Does hypothesis testing procedure attain the advertised level of significance or size (e.g. = 0.05)?  - What power is possible against different alternatives to the null hypothesis? Do different test procedures deliver different power? | |
| Properties of estimators | Monte Carlo simulation approximation | - Generate S independent data sets under the conditions of interest  - Compute the numerical value of the estimators  - If S large enough, summart statistics should a good approximations to the true properties of the estimator under the conditions of interest | |
| Checking the coverage probability of confidence interval |  | |
| Properties of hypothesis tests | Testing size/ level of test | Generate data under  Calculate the proportion of rejections of  Approximate the true probability of rejecting when it is true.  Proportion should be | |
| Evaluate power | Generate data under  Calculate the proportion of rejections of  Approximate the true probability of rejecting when it is false. | |
| Bootstrap Method | | | |
| Bootstrap distribution | Empiritcal distribution |  | |
| Empirical cumulative distribution |  | |
| Method | Estimate distribution function | Generate bootstrap replicate into  Calculate bootstrap estimate of | |
| Statistics | Standard Error | where | |
| Bias |  | |
| Confidence interval | Basic  Percentile  where is the sample quantitle from the empirical cdf of the replicates | |
| Numerical methods in R | | | |
| Root-finding | One-dimension | e.g. solve for y in equation | |
| Integration | One-variable | e.g. solve for the integral | |
| Maximum Likelihood Method | Concept | A method to estimate the parametes of a distribution, the parameter should maximizes the likelihood function  Assuming independent and identically distributed random values  Since population parameter is unknown, we estimate with  In reality the product function is difficult to differentiate, so we take log of the function | |
| Method1: finding which maximizes | Newton’s method to find the value which maximises the function | |
| Method2: finding the first order condition |  | |
| Optimization | One-dimension | e.g. Maximize the fuction | |
| Two-dimension | e.g. Find the maximum likelihood estimator of and  or consider the log-likelihood function | |
| Regression Analysis | | | |
| Correlation | Pearson Correlation Coefficient  - Indicates the strength of linear relationship between two variabels |  | |
| Sample regression | Linear model establish a relationship between two variables | where | |
| Assumptions | - ’s are independent  - ’s follow normal distribution with given parameters  Study residuals  - ’s have the same variances  Study residuals  - Linear relationship exist between Y and X  Test using scatter plot | |
| Prediction | Case1: values in the dataset  Case2: values not within the dataset | |
| Transformation | - Quadratic term to the regression line  - Log transformation | |
| Bonus | | | |
| Machine Learning | Extra | https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/ | |

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated 