I.2.2 - Tests of independence.

* Chi-square test.
* Fishers exact test.
* Cochran-Mantel-Haenszel test.

I.2.3. - Measures of association :-

* assocstats().

I.3.1. Types of correlation

* pearson, * spearman, * kendell, cor().
* partial correlation, pcor().

I.3.2 :- Testing correlations for significance:

* ~~[illegible]~~ cor.test().
* cor.test().
* pcor.test().

7.4. T-Test

7.4.1. Independent T-Test.

* t.test()

7.4.2. Dependent T-Test

t.test(y1, y2, paired=TRUE).

7.5. Nonparametric tests

7.5.1. Comparing 2 groups.

* Wilcox.test().
* wilcox.test(?, , Paired=TRUE).

7.5.2. Comparing more than 2 groups:-

* kruskal.test()
* friedman.test().

---

8. Regression.

8.2. Fitting regression models with lm().

* lm().
* Table 8.2, symbols commonly used in ~~Regression~~ R formulas.
* summary(), coefficients(), confint(), fitted(), residuals(), anova(), vcov(), AIC().

plot() - generate diagnostic plots for evaluating fit of model.

vcov() - lists covariance matrix for model parameters.

8.3. Regression diagnostics

* plot(fit).
* qqPlot() - Quantile comparison plot().
* durbinWatsonTest() - for auto correlated errors
* ~~crPlus~~ crPlots() - component plus residual plots.

*ncvTest() - Score test for nonconstant error variance.

*. spreadLevelPlot()

* outlierTest() - Bonferroni outlier test.

*. avPlots() - added variable plots.

* influencePlot() - regression influence plots.

*vif() - variation inflation factors.

Normality

*qqPlot()

Independence of errors

* durbinWatsonTest().

Linearity:

*crPlots().

Homoscedasticity - (constant error variance).

*ncvTest().

*spreadLevelPlot().

8.3.3.

library(gvlma).

variable ← gvlma(fit).

* performs a global validation of linear model assumptions as well as separate evaluations of skewness, kurtosis, and heteroscedasticity.

8.3.4. Multicollinearity, detection

* vif()
* $\sqrt{vif} > 2$, indicates multicollinearity.

8.4.1: Outliers.

* outlierTest().

8.4.2: High leverage points: (hat statistics).

8.4.3: Influential observations:

* Cook's distance.
* avPlots(), * influencePlot().

8.5.2: Transforming variables:

(1) When model violates normality assumption.

* powerTransform().

(2) When model violates linearity assumption.

* boxTidwell().

(3) When model violates homoscedasticity.

* spreadLevelPlot().

Comparing Models:- 8.6.1

* anova().

* AIC().

Variable selection. 8.6.2.

(1) Stepwise Regression.

* stepAIC().

2) All subsets Regression

* regsubsets(). in leap package
* subsets() in car package.

8.7.1. Cross validation.

8.7.2. Relative importance.

9. Analysis of variance :-

9.2.1 :- aov() function.

~~(9.2.2 :)~~

* Table 9.5. Formulas for common research designs.

9.2.2 : The order of formula terms.

9.3. One-way Anova

* aov (y ~ x1).
* plot means ( ) - plot group means and confidence interval.

9.3.1 : Multiple comparisions

* TukeyHSD ( ) - test of all pairwise differences between groups.
* glht ( ) in multcomp package. means

9.3.2 :- Assessing test assumptions :-

* Q-Q Plot - to assess normality assumption.
* bartlett.test ( ) - For homogeneity of variance. (equal variance in each group.

Other tests for homoscedasticity of variance.

* fligner.test().
* hov() in HH package.

9.4. One way Ancova

* aov()
* effect(), in effects library.
* glht() in multcomp library.

9.4.2:- Visualizing the results.

* ancova(), in HH library.

9.4.1:- Assessing test assumption.

* homogeneity of regression slopes.

aov(weight ~ gestfine * dose).

↓ Quantitative ↓ Qualitative.

9.5. Two way factorial Anova:-

* interaction.plot() - to display interaction in two-way Anova.

* plotmeans().
* interaction2wt().

aov(len ~ supp * dose)

Qualitative

9.6. Repeated measures ANOVA:-

* shows anova with one between and within groups factors.
* interaction.plot().

aov(uptake ~ conc * Type + Error(Plant / (conc)).

* boxplot().
* reshape() package
* sphericity - variances of differences b/w any 2 levels of within group factors are equal.

9.7. Manova.

* y ← cbind(calories, fat, sugars)
* manova (y ~ shelf).
* summary.aov (fit) - prints univariate results.
* TukeyHSD.

9.7.1: Assessing test assumptions:-

Two assumption
(1). multivariate normality. (← Q-Q plot)
(2). homogeneity of variance-covariance matrices.

* mahalanobis. } for 1st & assumption.
* qqplot()
* identify().

* Box's M test. - No R function for this.
* aq.plot() in mvoutlier package for multivariate outlier.

9.7.2 Robust Manova

~~* Wilks.test() in rrcov package.~~
~~* Non-parametric Manova.~~

* Robust Manova, one-way Manova.
  Wilks.test() in rrcov package.

* Nonparametric manova,
  adonis() in vegan package

9.8. Anova as Regression

Table 9.6. Built in contrasts.

- Contr.helmert
- Contr.poly.
- Contr.sum.
- Contr.treatment
- Contr.SAS.

- contrasts().
- options().

10. Power Analysis:-

- pwr package.

Table 10.1 pwr package:

- pwr.2p.test()
- pwr.2p2n.test()
- pwr.anova.test()
- pwr.chisq.test()
- pwr.f2.test()
- pwr.p.test()
- pwr.r.test()
- pwr.t.test()
- pwr.t2n.test().

10.2.1:- t-tests

- pwr.t.test(n=, d=, sig.level=, power=, type=, alternative

10.2.2:- Anova:-

- pwr.anova.test(k=, n=, f=, sig.level=, power=).

10.2.3. Correlations:-

~~[pwr.r.test]~~

- pwr.r.test(n=, r=, sig.level=, power=, alternative=).

10.2.4. Linear models:-

pwr.f2.test(u=, v=, f2=, sig.level=, power=).

10.2.5. Tests of proportions:

- pwr.2p.test(h=, n=, sig.level=, power=).

- ES.h( ).

10.2.6. Chi-Square tests:-

- pwr.chisq.test(w=, N=, df=, sig.level=, power=).

- ES.w2( ).

10.2.7. Choosing an appropriate effect size in novel situations:-

- Listing 10.1. - sample sizes for detecting significant effects in a one-way Anova.

11. Intermediate graphs:-

11.1. Scatterplots:-

- plot()
- lowess(), loess().
- scatterplot() in car package.

11.1.1:- Scatterplot matrices:-

- pairs().
- scatterplotMatrix().
- There are more functions mentioned.

11.1.2. High density scatterplots:-

11.1.3. 3D Scatter plots:-

11.1.4. Spinning 3D scatter plots:-

11.1.5. Bubbleplots.

- symbols()

11.2. Line charts:- plot(), lines().

- type = "b"

Line chart options. Table 11.1.

p - Points only.
L - Lines only.
o - Overplotted points.
b, c - points joined by lines.
s, S - stair steps.
h - Histogram-like vertical lines.
n. - Doesn't produce any points.

11.3. Corrgrams:

- corrgram().
- When order = TRUE, the variable are reordered using principal component Analysis of the correlation matrix.
- corrgram(x, order= , panel= , text.panel=

diag.panel =).

- Table 11.2 panel options for corrgram() function.
  - colorRampPalette(), col.regions option.

4. Mosaic plots:-
   - mosaic() in vcd library.

12. Resampling statistics and Bootstrapings

12.1. Permutation tests:-

- Coin package.
- ImPerm package.

12.2. Permutation tests with coin package

- Table 12.2 coin functions providing permutation tests alternatives to traditional tests.

- oneway_test(y~A)
- wilcox_test()
- kruskal_test()
- chisq_test().
- cmh_test()
- lbl_test()
- spearman_test().
- friedman_test()
- wilcoxsign_test().

12.2.1. Independent two sample and k-sample tests:-

- one-way_test().
- transform().
- wilcox_test().

12.2.2. Independence in contingency tables:-

- chisq_test()
- cmh_test().
- lbl_test().

12.2.3. Independence blw numeric variables.-

- spearman_test().

12.2.4. Dependent two sample and k-sample tests:-

- wilcoxonsign_test().

12.3. Permutation tests with lmPerm package:-

- lmp (lm() modified)
- aovp (aov() modified)
- perm= Exact, Prob, SPR. Exact.

12.3.1. Simple and polynomial regression.

12.3.2. Multiple regression.

12.3.3. One-way Anova and Ancova:-

12.3.4. Two-way Anova:-

12.5. Bootstrapping:-

- boot package.
- boot().
- boot.ci()

12.6. Bootstrapping with ~~(Statistic)~~ Boot package

12.6.1. Bootstrapping a single statistic.

12.6.2. Bootstrapping several statistic.

13. Generalized linear models:-

13.1.1. glm() function

- glm(formula, family=family(link=functio
  data=).

- Table 13.1. glm parameters:-

Family	Default link function.
• binomial	(link="logit").
• gaussian	="identity")
• gamma	="inverse")
• inverse.gaussian	="1/mu^2")
• poisson	="log")
• quasi	="identity", variance=constant)
quasibinomial	="logit")
quasipoisson	="log").

13.1.3. Model fit and regression diagnostics:-

- hatvalues()
- rstudent()
- cooks.distance().
- influencePlot().

13.2. Logistic Regression:-

- To compare 2 nested models, use anova().
- In generalized linear models, you'll want a chi-square version of this test.
- anova(model1, model2, test="Chisq").

13.2.1. Interpreting the model parameters:-

- coef(fit.reduced).
- exp(coef(fit.reduced)).
- exp(confint(fit.reduced)).

13.2.2 :- Assessing the impact of predictors on the probability of an outcome :-

13.2.3. Overdispersion :-

- Overdispersion occurs when the observed variance of response variable is larger than what would be expected from a binomial distribution.
- Over dispersion can lead to distorted test standards errors and inaccurate tests of significance.
- $\phi = \frac{\text{Residual deviance}}{\text{Residual df.}}$
- deviance ( ).

Test for overdispersion.

- you fit the model twice, but in first instance you use family = "binomial" ,in second instance family = "quasibinomial"
- The glm object returned in first case is "fit" and in second case is "fit.od".
- pchisq(summary(fit.od)$dispersion * fit$df.residual, fit$df.residual, lower=F).
- H_0: $\phi = 1$
  H_1: $\phi = 0$.

13.2.4. Extensions:-

1. Robust logistic Regression

- glmRob() in robust package.
- fitting logistic regression models to data containing outliers and influential observations.

2. Multinomial logistic Regression.

- If the response variable has more than two unordered categories.
- mlogit() in mlogit package.

3. Ordinal logistic Regression:-

- respose variable is a set of ordered categories.
- lrm() in rms package.

13.3. Poisson regression:-

- The qcc package provides a test for overdispersion in poisson case.
- qcc.overdispersion.test(), in qcc package.

13.3.1. Interpreting model parameters.

13.3.2. Overdispersion.

13.3.3. Extensions.

1. Poisson regression with varying time periods.
2. Zero inflated poisson regression.
   - zeroinfl in pscl package.
3. Robust poisson regression:-
   - glmRob() in robust package.

14. Principal components and factor analysis:-

14.1. Principal components and factor analysis in R:-

- princomp().
- factanal().

} In base R

Table 14.1. Useful factor analytic functions in psych package:-

- principal()
- fa().
- fa.parallel().
- factor.plot()
- fa.diagram()
- scree().

14.2. Principal components.

14.2.1. Selecting no. of components to extract:-

- Kaiser-Harris criterion. - retain components with eigen values greater than one.
- Cartell Scree test - eigen values are plotted against their component numbers.
- run simulations, extracting eigen values from random data matrices of same size. as original matrix. If an eigen value based on real data is > than average corresponding eigen values from a set of random matrices, that component is retained.
- fa. parallel ( ).

14.2.2. Extracting principal components:-

- principal (r, nfactors=, rotate=, scores=).

14.2.3. Rotating principal components:-

- varimax.

14.2.4. Obtaining principal component scores:-

14.3. Explanatory Factor Analysis:-

14.3.1. Deciding How many factors to extract:-

- For EFA, the Kaiser-Harris criterion is eigen values greater than 0.

14.3.2. Extracting common factors:-

- fa (r, nfactors= , n.obs= , rotate= , scores= , fm=).

fm - factoring method (minres default).

- "ml" - maximum likelihood.
- "pa" - iterated principal axis.
- "uls" - weighted least squares.
- "gls" - generalized weighted least squares
- "minres" - minimum residuals.

14.3.3. Rotating factors:-

- varimax. - orthogonal
- promax - oblique.
- factor.plot()
- fa.diagram().

14.3.4:- Factor scores:-

14.3.5: Other EFA-related packages:-

14.4. ~~Confirmatory~~ Other latent variable models:

- Confirmatory Factor Analysis.
- Structural equation modelling. - sem, OpenMx, lavaan.
- ltm - latent models.
- Latent class models - FlexMix, lcmm, randomLCA, and poLCA.
- lcda - latent class discriment analysis.
- lsa - latent semantic analysis.
- ca - simple and ~~multidimensional~~ multiple correspondence analysis.
- multidimensional scaling. - cmdscale()
  - isoMDS() in MASS.

15. Time series

Table 15.1 Functions for time series analysis:-

Function	package.
• ts()	stats
• start()	stats
• end()	stats
• frequency()	stats
• window().	stats.
• ma().	forecast
• stl()	stats
• monthplot()	stats
• seasonplot()	forecast.
• HoltWinters()	stats.
• forecast()	forecast

Function	package
• accuracy.()	forecast
• ets()	forecast
• lag()	stats
• Acf().	stats
• Pacf()	stats
• diff().	base.
• ndiff()	forecast.
• adf.test()	tseries.
• arima()	stats
• Box.test()	stats
• bds.test.()	tseries
• auto.arima()	forecast

- ts(data, start= , end= , frequency)
- start(tsales)
- end(tsales)
- frequency(tsales).

- tsales.subset ← window(tsales, start= , end=).

15.2. Smoothing and seasonal decomposition

15.2.1 Smoothing with simple moving average:-

- centered moving average.
- SMA() in TTR.
- rollmean() in zoo,
- ma() in forecast.

15.2.2. Seasonal decomposition:-

- trend component.
- seasonal component.

- irregular (error) component.

- stl(ts, s.window= , t.window=).

ts - time series to be decomposed.

s.window - controls how fast the seasonal effects change over time.

t.window - how fast the trend can change over time.

- s.window = "periodic" - forces seasonal effects to be identical over years.

- stl() can only handle additive models. But we can handle multiplicative models using log.

- monthplot ( ).

- seasonplot ( ) in forecast.

15.3. Exponential forecasting models:-

- HoltWinter()
- ets() in forecast.
- ets(ts, model="ZZZ").

Table. 15.3. Functions for fitting simple, double, and triple exponential forecasting models:

Type.	parameters fit	Functions.
simple	level	ets(ts, model = "ANN") ses(ts).
double	level & slope	ets(ts, model = "AAN") holt(ts).
triple	level, slope, seasonal.	ets(ts, model = "AAA") hw(ts).

15.3.1. Simple exponential smoothing :-

- forecast() is used to predict time series k steps into the future.
- forecast(fit, k).
- accuracy(fit).

~~5.3.2. Holt and Holt-Winters Exponential~~

- Model equation:

$$L_t = \alpha Y_t + (1-\alpha) L_{t-1}$$

- Y_t is actual observation at time
- α is smoothing parameter,
- L_t is previous level (smoothed value) or average.

15.3.2. Holt and Holt winters exponential smoothing:-

- Holt exponential smoothing model equations.

1. Level equation.

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + T_{t-1}).$$

2. Trend eq:-

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}.$$

3. Forecast eq:-

$$\hat{Y}_{t+k} = L_t + T_t \times k.$$

$\hat{Y}_{t+k}$ is forecast for k periods ahead.

α: Controls smoothing of level

β: Controls smoothing of trend.

Holt-winters exponential smoothing:-

- Model equations:

Additive seasonality.

- Level eq:

$$L_t = \alpha (Y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + T_{t-1})$$

- Trend eq:

$$T_t = \beta (L_t - L_{t-1}) + (1-\beta) T_{t-1}$$

- Seasonality eq:

$$S_t = \gamma (Y_t - L_t) + (1-\gamma) S_{t-s}$$

- Forecast eq:

$$\hat{Y}_{t+k} = L_t + T_t * k + S_{t+k-s}.$$

15.3.3. The ets() function and automated forecasting:

- ets(AirPassengers, model = "MAM")
- hw(AirPassengers, seasonal = "multiplicative")
- et(JohnsonJohnson) ⇒ automatic select best fitting model.

15.4. ARIMA forecasting models:-

15.4.1. Prerequisite concepts:-

- lag(ts, k), k = no. of lags.
- Autocorrelation function plot with
  (1) acf() in stats.
  (2) Acf() on forecast
- partial autocorrelation plot with.
  (1) pacf() in stats.
  (2) Pacf() in forecast.

- ARIMA models are designed to fit stationary time series.
- In stationary ts, the statistical properties dont change over time.
- The mean and variance are constant.
- The autocorrelations of any lag k dont change over time.
- differencing, $Y_{t-1} - Y_t$.
- differencing once, removes linear trend
- differencing twice, removes quadratic trend
- differencing thrice remove cubic trend
- diff(ts, differences = d).
- ndiffs() in forecast → can help to determine best value of d.
- Augmented Dickey-Fuller (ADF) test - to evaluate stationarity assumption.

→ adf.test() in tseries

- In summary,

(1) ACF and PACF plots to determine parameters of ARIMA.

(2). Transformation and differencing used to help achieve stationarity.

15.4.2. ARMA and ARIMA models:-

ARMA(p,q)

$$Y_t = \mu + B_1 Y_{t-1} + B_2 Y_{t-2} + \dots + B_p Y_{t-p} - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} \dots - \theta_q \varepsilon_{t-q} + \varepsilon_t.$$

from past p values and q residuals

- ACF plot used to identify q parameter
- PACF plot for p parameter.

- Evaluating model fit:-
  - qqnorm() & qqline().
  - Box.test(fit$residuals, type="Ljung-Box")
  - If model is appropriate, residuals should be normally and independently distributed. (no relationship between them) or (no autocorrelation).
  - Box.test() provides a test for autocorrelation.

15.4.3. Automated ARIMA forecasting:-

- auto.arima() in forecast.