**ITA 0443**

**STATISTICS WITH R PROGRAMMING FOR REAL TIME PROBLEM**

DAY 4 – LAB ASSESSMENT

Reg No:192011486

Name:V.PENCHALA REDDY

GITHUB LINK : [penchala720/ITA0443-R-Programming (github.com)](https://github.com/penchala720/ITA0443-R-Programming)

1.Randomly Sample the iris dataset such as 80% data for training and 20% for test and   create Logistics regression with train data, use species as target and petals width and  
length as feature variables , Predict the probability of the model using test data,  Create Confusion matrix for above test model

2. (i)Write suitable R code to compute the mean, median ,mode of the following values  
            c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

programm and output:

x<-c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

> y<-mean(x)

> print(y)

[1] 60

> y<-mode(y)

> print(y)

[1] "numeric"

> y<-median(x)

> print(y)

[1] 70

> y<-mode(x)

> print(y)

[1] "numeric"

(ii) Write R code to find 2nd  highest and 3rd Lowest value of above problem.

programm and output:

vec = c(90, 50, 70, 80, 70, 60, 20, 30, 80, 90, 20)

> sorted\_vec = sort(vec, decreasing = TRUE)

> second\_highest = sorted\_vec[2]

> sorted\_vec = sort(vec)

> third\_lowest = sorted\_vec[3]

> cat("2nd highest value:", second\_highest, "\n")

2nd highest value: 90

> cat("3rd lowest value:", third\_lowest, "\n")

3rd lowest value: 30

3. Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months:  
• Ozone: mean ozone concentration (ppb), • Solar.R: solar radiation (Langley),  
• Wind: average wind speed (mph), • Temp: maximum daily temperature in degrees Fahrenheit,  
• Month: numeric month (May=5, June=6, and so on),• Day: numeric day of the month (1 -4).

 i. Compute the mean temperature(don’t use build in function)

**programm:**

data("airquality")

> mean\_temp <- sum(airquality$Temp) / nrow(airquality)

**output:**

1

170  
ii.Extract the first five rows from airquality.  
iii.Extract all columns from airquality except Temp and Wind  
iv.Which was the coldest day during the period?  
v.How many days was the wind speed greater than 17 mph?

4. (i)Get the Summary Statistics of air quality dataset  
 (ii)Melt airquality data set and display as a long – format data?  
 (iii)Melt airquality data and specify month and day to be “ID variables”?  
  (iv)Cast the molten airquality data set with respect to month and date features  
  (v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

5.(i) Find any missing values(na) in features and drop the missing values if its less than 10%  
 else replace that with  mean of that feature.  
    (ii) Apply a linear regression algorithm using Least Squares Method on “Ozone” and “Solar.R”  
    (iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model

6. Load dataset named ChickWeight,   
 ( i).Order the data frame, in ascending order by feature name “weight” grouped by   feature   
 “diet” and Extract the last 6 records from order data frame.  
  (ii).a Perform melting function based on “Chick", "Time", "Diet"   features as ID variables  
  b. Perform cast function to display the mean value of weight grouped by Diet   
  c. Perform cast function to display the mode of weight grouped by Diet

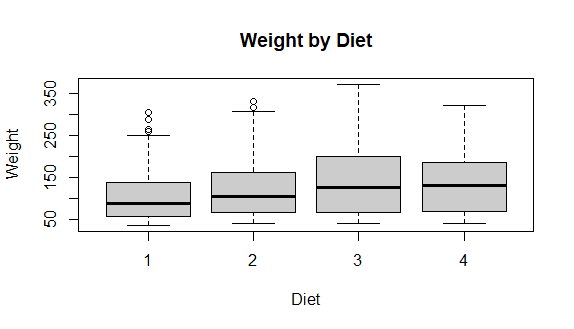
7. a.  Create Box plot for “weight” grouped by “Diet”

**programm:**

> boxplot(weight ~ Diet, data = ChickWeight, main = "Weight by Diet", xlab = "Diet", ylab = "Weight")

>

**output:**

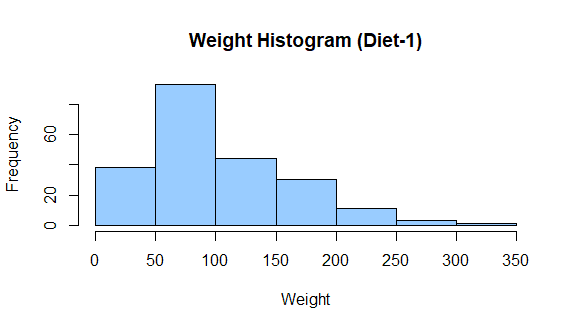


          b. Create a Histogram for “weight” features belong to Diet- 1 category

**programm:**

hist(ChickWeight$weight[ChickWeight$Diet == 1], main = "Weight Histogram (Diet-1)", xlab = "Weight", col = "lightblue")

**output:**



>   
          c.  Create Scatter plot for “ weight” vs “Time” grouped by Diet  
  
8.   a. Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as  as predictor variables

**programm:**

data("ChickWeight")

> model <- lm(weight ~ Time + Diet, data = ChickWeight)

> new\_data <- data.frame(Time = 10, Diet = 1)

> prediction <- predict(model, newdata = new\_data)

Error: variable 'Diet' was fitted with type "factor" but type "numeric" was supplied

In addition: Warning message:

In model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels) :

variable 'Diet' is not a factor

> print(prediction)

**output:**

1

1392.46   
          b. Predict weight for Time=10 and Diet=1  
           c. Find the error in model for same

**programm:**

error <- residuals(model)

> print(error)

**output:**

1 2 3 4 5

31.07560890 22.57462541 13.07364193 0.57265844 -4.92832504

6 7 8 9 10

-5.42930852 -9.93029201 -8.43127549 -1.93225898 2.56675754

11 12 13 14 15

13.06577405 10.31528231 29.07560890 20.57462541 12.07364193

16 17 18 19 20

8.57265844 3.07167496 4.57069148 6.06970799 4.56872451

21 22 23 24 25

11.06774102 18.56675754 23.06577405 20.31528231 32.07560890

26 27 28 29 30

10.57462541 9.07364193 3.57265844 3.07167496 0.57069148

31 32 33 34 35

-0.93029201 4.56872451 12.06774102 18.56675754 12.06577405

36 37 38 39 40

7.31528231 31.07560890 20.57462541 10.07364193 3.57265844

41 42 43 44 45

-6.92832504 -11.42930852 -13.93029201 -25.43127549 -14.93225898

46 47 48 49 50

-14.43324246 -25.93422595 -37.68471769 30.07560890 13.57462541

51 52 53 54 55

2.07364193 -3.42734156 -1.92832504 7.57069148 25.06970799

56 57 58 59 60

30.56872451 46.06774102 30.56675754 34.06577405 28.31528231

61 62 63 64 65

30.07560890 20.57462541 13.07364193 10.57265844 16.07167496

66 67 68 69 70

25.57069148 25.06970799 14.56872451 4.06774102 -8.43324246

71 72 73 74 75

-25.93422595 -37.68471769 30.07560890 20.57462541 11.07364193

76 77 78 79 80

7.57265844 8.07167496 13.57069148 30.06970799 40.56872451

81 82 83 84 85

67.06774102 81.56675754 102.06577405 110.31528231 31.07560890

86 87 88 89 90

21.57462541 15.07364193 7.57265844 3.07167496 -5.42930852

91 92 93 94 95

-5.93029201 -17.43127549 -24.93225898 -34.43324246 -60.93422595

96 97 98 99 100

31.07560890 22.57462541 13.07364193 4.57265844 4.07167496

101 102 103 104 105

-2.42930852 -25.93029201 -41.43127549 -57.93225898 -68.43324246

106 107 108 109 110

-85.93422595 -96.68471769 30.07560890 15.57462541 6.07364193

111 112 113 114 115

-0.42734156 -6.92832504 -17.42930852 -26.93029201 -37.43127549

116 117 118 119 120

-49.93225898 -56.43324246 -65.93422595 -70.68471769 32.07560890

121 122 123 124 125

22.57462541 17.07364193 20.57265844 31.07167496 40.57069148

126 127 128 129 130

52.06970799 43.56872451 31.06774102 15.56675754 -4.93422595

131 132 133 134 135

-19.68471769 30.07560890 20.57462541 10.07364193 -1.42734156

136 137 138 139 140

-8.92832504 -10.42930852 3.06970799 1.56872451 11.06774102

141 142 143 144 145

16.56675754 9.06577405 10.31528231 30.07560890 19.57462541

146 147 148 149 150

7.07364193 -3.42734156 -15.92832504 -31.42930852 -44.93029201

151 152 153 154 155

-63.43127549 -79.93225898 -87.43324246 -94.93422595 -98.68471769

156 157 158 159 160

30.07560890 20.57462541 16.07364193 15.57265844 20.07167496

161 162 163 164 165

29.57069148 48.06970799 58.56872451 76.06774102 79.56675754

166 167 168 169 170

73.06577405 71.31528231 30.07560890 20.57462541 10.07364193

171 172 173 174 175

0.57265844 -12.92832504 -30.42930852 -48.93029201 -65.43127549

176 177 178 179 180

30.07560890 16.57462541 3.07364193 -12.42734156 -23.92832504

181 182 183 184 185

-47.42930852 -61.93029201 31.07560890 22.57462541 15.07364193

186 187 188 189 190

8.57265844 2.07167496 -9.42930852 -17.93029201 -30.43127549

191 192 193 194 195

-37.93225898 -45.43324246 -52.93422595 -52.68471769 28.07560890

196 197 198 199 200

6.57462541 32.07560890 19.57462541 9.07364193 -1.42734156

201 202 203 204 205

-15.92832504 -27.42930852 -33.93029201 -45.43127549 -44.93225898

206 207 208 209 210

-48.43324246 -41.93422595 -37.68471769 30.07560890 18.57462541

211 212 213 214 215

8.07364193 -5.42734156 -15.92832504 -25.42930852 -38.93029201

216 217 218 219 220

-44.43127549 -52.93225898 -61.43324246 -70.93422595 -77.68471769

221 222 223 224 225

12.90953485 5.40855137 -0.09243212 6.40658440 27.90560091

226 227 228 229 230

48.40461743 84.90363395 90.40265046 107.90166698 122.40068349

231 232 233 234 235

115.89970001 120.14920827 13.90953485 10.40855137 1.90756788

236 237 238 239 240

-2.59341560 -7.09439909 -19.59538257 -24.09636605 -38.59734954

241 242 243 244 245

-36.09833302 -36.59931651 -38.10029999 -43.85079173 15.90953485

246 247 248 249 250

7.40855137 -1.09243212 -6.59341560 -7.09439909 -11.59538257

251 252 253 254 255

-5.09636605 -14.59734954 -22.09833302 -21.59931651 -32.10029999

256 257 258 259 260

-35.85079173 14.90953485 7.40855137 -4.09243212 -5.59341560

261 262 263 264 265

-31.09439909 -46.59538257 -62.09636605 -78.59734954 -95.09833302

266 267 268 269 270

-112.59931651 -126.10029999 -136.85079173 12.90953485 4.40855137

271 272 273 274 275

-0.09243212 -1.59341560 4.90560091 9.40461743 13.90363395

276 277 278 279 280

14.40265046 29.90166698 46.40068349 56.89970001 54.14920827

281 282 283 284 285

14.90953485 3.40855137 -5.09243212 -5.59341560 -4.09439909

286 287 288 289 290

-0.59538257 3.90363395 -2.59734954 1.90166698 20.40068349

291 292 293 294 295

33.89970001 40.14920827 11.90953485 1.40855137 -4.09243212

296 297 298 299 300

-6.59341560 -10.09439909 -14.59538257 -17.09636605 -26.59734954

301 302 303 304 305

-23.09833302 -21.59931651 -17.10029999 -18.85079173 11.90953485

306 307 308 309 310

1.40855137 -4.09243212 -6.59341560 -5.09439909 -0.59538257

311 312 313 314 315

12.90363395 6.40265046 16.90166698 22.40068349 9.89970001

316 317 318 319 320

22.14920827 11.90953485 3.40855137 -3.09243212 -5.59341560

321 322 323 324 325

-10.09439909 -8.59538257 1.90363395 0.40265046 19.90166698

326 327 328 329 330

45.40068349 76.89970001 98.14920827 14.90953485 3.40855137

331 332 333 334 335

-3.09243212 -7.59341560 -12.09439909 -16.59538257 -17.09636605

336 337 338 339 340

-27.59734954 -24.09833302 -33.59931651 -45.10029999 -60.85079173

341 342 343 344 345

-5.42379848 -11.92478197 -20.42576545 -26.92674893 -32.42773242

346 347 348 349 350

-32.92871590 -29.42969939 -31.93068287 -17.43166636 -0.93264984

351 352 353 354 355

12.56636667 24.81587493 -6.42379848 -15.92478197 -17.42576545

356 357 358 359 360

-17.92674893 -10.42773242 -5.92871590 6.57030061 9.06931713

361 362 363 364 365

33.56833364 58.06735016 68.56636667 73.81587493 -8.42379848

366 367 368 369 370

-14.92478197 -19.42576545 -22.92674893 -21.42773242 -23.92871590

371 372 373 374 375

-15.42969939 -25.93068287 -36.43166636 -58.93264984 -66.43363333

376 377 378 379 380

-84.18412507 -6.42379848 -15.92478197 -19.42576545 -14.92674893

381 382 383 384 385

-10.42773242 -0.92871590 11.57030061 16.06931713 47.56833364

386 387 388 389 390

89.06735016 104.56636667 109.81587493 -6.42379848 -11.92478197

391 392 393 394 395

-18.42576545 -12.92674893 5.57226758 23.07128410 48.57030061

396 397 398 399 400

68.06931713 99.56833364 127.06735016 138.56636667 141.81587493

401 402 403 404 405

-8.42379848 -16.92478197 -21.42576545 -23.92674893 -19.42773242

406 407 408 409 410

-18.92871590 -7.42969939 -3.93068287 10.56833364 22.06735016

411 412 413 414 415

2.56636667 -11.18412507 -6.42379848 -16.92478197 -26.42576545

416 417 418 419 420

-31.92674893 -37.42773242 -51.92871590 -49.42969939 -57.93068287

421 422 423 424 425

-52.43166636 -47.93264984 -53.43363333 -53.18412507 -6.42379848

426 427 428 429 430

-15.92478197 -21.42576545 -25.92674893 -19.42773242 -25.92871590

431 432 433 434 435

-24.42969939 -15.93068287 4.56833364 27.06735016 57.56636667

436 437 438 439 440

58.81587493 -5.42379848 -14.92478197 -21.42576545 -21.92674893

441 442 443 444 445

-28.42773242 -25.92871590 -22.42969939 -23.93068287 -17.43166636

446 447 448 449 450

9.06735016 27.56636667 40.81587493 -6.42379848 -9.92478197

451 452 453 454 455

-16.42576545 -20.92674893 -16.42773242 -14.92871590 1.57030061

456 457 458 459 460

12.06931713 27.56833364 57.06735016 72.56636667 89.81587493

461 462 463 464 465

0.84215272 -7.65883076 -10.15981425 -8.66079773 -8.16178122

466 467 468 469 470

-4.66276470 8.83625181 -10.66473167 -6.16571516 -14.66669864

471 472 473 474 475

-17.16768213 -20.91817387 0.84215272 -9.65883076 -13.15981425

476 477 478 479 480

-9.66079773 -8.16178122 -2.66276470 13.83625181 10.33526833

481 482 483 484 485

22.83428484 35.33330136 52.83231787 56.08182613 0.84215272

486 487 488 489 490

-3.65883076 -7.15981425 2.33920227 19.83821878 28.33723530

491 492 493 494 495

37.83625181 24.33526833 15.83428484 -0.66669864 -17.16768213

496 497 498 499 500

-24.91817387 0.84215272 -7.65883076 -11.15981425 -7.66079773

501 502 503 504 505

-8.16178122 -10.66276470 -19.16374819 -25.66473167 -36.16571516

506 507 508 509 510

-52.66669864 -0.15784728 -8.65883076 -15.15981425 -15.66079773

511 512 513 514 515

-13.16178122 -11.66276470 -11.16374819 -22.66473167 -34.16571516

516 517 518 519 520

-24.66669864 -19.16768213 -28.91817387 -1.15784728 -6.65883076

521 522 523 524 525

-14.15981425 -11.66079773 -10.16178122 -8.66276470 -2.16374819

526 527 528 529 530

-7.66473167 -8.16571516 11.33330136 14.83231787 13.08182613

531 532 533 534 535

-0.15784728 -5.65883076 -10.15981425 -14.66079773 -11.16178122

536 537 538 539 540

-5.66276470 1.83625181 -6.66473167 -13.16571516 -13.66669864

541 542 543 544 545

-6.16768213 -19.91817387 -2.15784728 -8.65883076 -14.15981425

546 547 548 549 550

-13.66079773 -7.16178122 -3.66276470 7.83625181 6.33526833

551 552 553 554 555

40.83428484 62.33330136 86.83231787 97.08182613 -1.15784728

556 557 558 559 560

-5.65883076 -12.15981425 -8.66079773 -3.16178122 -0.66276470

561 562 563 564 565

5.83625181 2.33526833 2.83428484 4.33330136 16.83231787

566 567 568 569 570

12.08182613 -0.15784728 -4.65883076 -9.15981425 -9.66079773

571 572 573 574 575

-6.16178122 -6.66276470 8.83625181 11.33526833 23.83428484

576 577 578

35.33330136 47.83231787 39.08182613

**>**

9 .For this exercise, use the (built-in) dataset Titanic.  
    a. Draw a Bar chart to show details of “Survived” on the Titanic based on passenger Class  
    b. Modify the above plot based on gender of people who survived  
   c. Draw histogram plot to show distribution of feature “Age”

10. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area.   
 (i) a. Explore the summary of Data set, like number of Features and its type. Find the number         of records for each feature. Print the statistical feature of data

**programm and output:**

View(USArrests)

> data("USArrests")

> str(USArrests)

'data.frame': 50 obs. of 4 variables:

$ Murder : num 13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...

$ Assault : int 236 263 294 190 276 204 110 238 335 211 ...

$ UrbanPop: int 58 48 80 50 91 78 77 72 80 60 ...

$ Rape : num 21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...

> nrow(USArrests)

[1] 50

> summary(USArrests)

Murder Assault UrbanPop Rape

Min. : 0.800 Min. : 45.0 Min. :32.00 Min. : 7.30

1st Qu.: 4.075 1st Qu.:109.0 1st Qu.:54.50 1st Qu.:15.07

Median : 7.250 Median :159.0 Median :66.00 Median :20.10

Mean : 7.788 Mean :170.8 Mean :65.54 Mean :21.23

3rd Qu.:11.250 3rd Qu.:249.0 3rd Qu.:77.75 3rd Qu.:26.18

Max. :17.400 Max. :337.0 Max. :91.00 Max. :46.00

>   
         b. Print the state which saw the largest total number of rape

**programm:**

rape\_arrests <- USArrests[which.max(USArrests$Rape),]

> rownames(rape\_arrests)

**output:**

[1] "Nevada"

>   
         c. Print the states with the max & min crime rates for murder

**programm:**

murder\_arrests <- USArrests[, "Murder"]

> max\_murder\_arrests <- rownames(USArrests[which.max(murder\_arrests),])

> min\_murder\_arrests <- rownames(USArrests[which.min(murder\_arrests),])

> c(max\_murder\_arrests, min\_murder\_arrests)

**output:**

[1] "Georgia" "North Dakota"  
    (ii).a. Find the correlation among the features

**programm and output:**

cor(USArrests)

Murder Assault UrbanPop Rape

Murder 1.00000000 0.8018733 0.06957262 0.5635788

Assault 0.80187331 1.0000000 0.25887170 0.6652412

UrbanPop 0.06957262 0.2588717 1.00000000 0.4113412

Rape 0.56357883 0.6652412 0.41134124 1.0000000  
     b. Print the states which have assault arrests more than median of the country

**programm:**

> median\_assault\_arrests <- median(USArrests$Assault)

> states\_with\_above\_median\_assault <- rownames(USArrests[USArrests$Assault > median\_assault\_arrests,])

> states\_with\_above\_median\_assault

**output:**

[1] "Alabama" "Alaska" "Arizona" "Arkansas"

[5] "California" "Colorado" "Delaware" "Florida"

[9] "Georgia" "Illinois" "Louisiana" "Maryland"

[13] "Michigan" "Mississippi" "Missouri" "Nevada"

[17] "New Mexico" "New York" "North Carolina" "Rhode Island"

[21] "South Carolina" "Tennessee" "Texas" "Wyoming"   
     c. Print the states are in the bottom 25% of murder

**programm:**

murder\_arrests\_bottom\_25 <- quantile(USArrests$Murder, 0.25)

> states\_in\_bottom\_25 <- rownames(USArrests[USArrests$Murder < murder\_arrests\_bottom\_25,])

> states\_in\_bottom\_25

**output:**

[1] "Connecticut" "Idaho" "Iowa" "Maine"

[5] "Minnesota" "New Hampshire" "North Dakota" "Rhode Island"

[9] "South Dakota" "Utah" "Vermont" "Washington"

[13] "Wisconsin"

>   
   (iii). a. Create a histogram and density plot of murder arrests by US stat

**programm:**

hist(USArrests$Murder)

> density(USArrests$Murder)

**output:**

Call:

density.default(x = USArrests$Murder)

Data: USArrests$Murder (50 obs.); Bandwidth 'bw' = 1.793

x y

Min. :-4.578 Min. :5.639e-05

1st Qu.: 2.261 1st Qu.:5.526e-03

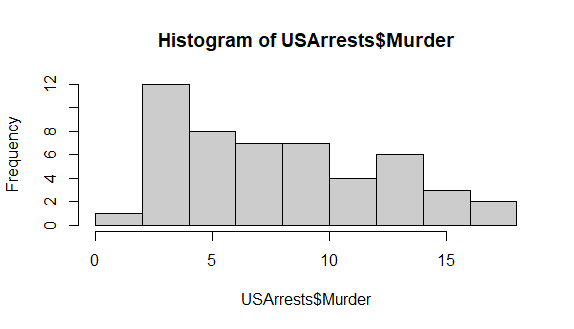
Median : 9.100 Median :3.592e-02

Mean : 9.100 Mean :3.652e-02

3rd Qu.:15.939 3rd Qu.:6.364e-02

Max. :22.778 Max. :7.908e-02

>



b. Create the plot that shows the relationship between murder arrest rate and  proportion

of the population that is urbanised by state. Then enrich the chart by adding assault

arrest rates (by colouring the points from blue (low) to red (high)).  
         c. Draw a bar graph to show the murder rate for each of the 50 states .

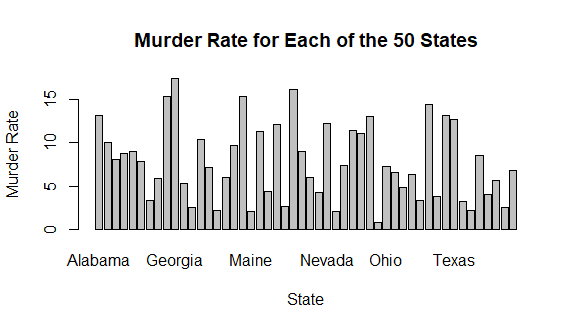
**programm:**

> barplot(USArrests$Murder, names.arg = rownames(USArrests),

+ main = "Murder Rate for Each of the 50 States",

+ xlab = "State", ylab = "Murder Rate")

**output:**



11. a. Create a data frame based on below table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Spends | 1000 | 4000 | 5000 | 4500 | 3000 | 4000 | 9000 | 11000 | 15000 | 12000 | 7000 | 3000 |
| Sales | 9914 | 40487 | 54324 | 50044 | 34719 | 42551 | 94871 | 118914 | 158484 | 131348 | 78504 | 36284 |

**programm:**

Month <- c(1:12)

> Spends <- c(1000, 4000, 5000, 4500, 3000, 4000, 9000, 11000, 15000, 12000, 7000, 3000)

> Sales <- c(9914, 40487, 54324, 50044, 34719, 42551, 94871, 118914, 158484, 131348, 78504, 36284)

> df <- data.frame(Month, Spends, Sales)

> print(df)

**output:**

Month Spends Sales

1 1 1000 9914

2 2 4000 40487

3 3 5000 54324

4 4 4500 50044

5 5 3000 34719

6 6 4000 42551

7 7 9000 94871

8 8 11000 118914

9 9 15000 158484

10 10 12000 131348

11 11 7000 78504

12 12 3000 36284

b. Create a regression model for that data frame table to show the amount of sales(Sales) based on the how much the company spends (Spends) in advertising

c. Predict the Sales if Spend=13500

**programm:**

predict(model, data.frame(Spends = 13500))

**output:**

1

144783.1

**Set 2**

1.(i) Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution.)

**programm:**

> letters\_sample <- sample(LETTERS, 20)

> letters\_factor <- factor(letters\_sample)

> letters\_levels <- levels(letters\_factor)[1:5]

> letters\_levels

**output:**

[1] "A" "B" "C" "D" "F"

(ii)Write R function to find the range of given vector. Range=Max-Min  
Sample input, C<-(9,8,7,6,5,4,3,2,1),  
output=8

**programm:**

find\_range <- function(vec) {

max\_val <- max(vec)

min\_val <- min(vec)

range <- max\_val - min\_val

return(range)

}

C <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)

result <- find\_range(C)

print(result)

**output:** 1

8  
  
   (iii)Wirte the R function to find the number of vowels in given string  
 Sample input c<- “matrix”,  output<-2

**programm:**count\_vowels <- function(str) {

vowels <- c("a", "e", "i", "o", "u", "A", "E", "I", "O", "U")

str\_vowels <- str[str %in% vowels]

return(nchar(str\_vowels))

}

c <- "matrix"

result <- count\_vowels(c)

print(result)

**output:**

2  
  
2.Load inbuild dataset “ChickWeight” in R  
(i) Explore the summary of Data set, like number of Features and its type. Fins the number of records for each features

**programm and output:**

data("ChickWeight")

> str(ChickWeight)

Classes ‘nfnGroupedData’, ‘nfGroupedData’, ‘groupedData’ and 'data.frame': 578 obs. of 4 variables:

$ weight: num 42 51 59 64 76 93 106 125 149 171 ...

$ Time : num 0 2 4 6 8 10 12 14 16 18 ...

$ Chick : Ord.factor w/ 50 levels "18"<"16"<"15"<..: 15 15 15 15 15 15 15 15 15 15 ...

$ Diet : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...

- attr(\*, "formula")=Class 'formula' language weight ~ Time | Chick

.. ..- attr(\*, ".Environment")=<environment: R\_EmptyEnv>

- attr(\*, "outer")=Class 'formula' language ~Diet

.. ..- attr(\*, ".Environment")=<environment: R\_EmptyEnv>

- attr(\*, "labels")=List of 2

..$ x: chr "Time"

..$ y: chr "Body weight"

- attr(\*, "units")=List of 2

..$ x: chr "(days)"

..$ y: chr "(gm)"

> summary(ChickWeight)

weight Time Chick Diet

Min. : 35.0 Min. : 0.00 13 : 12 1:220

1st Qu.: 63.0 1st Qu.: 4.00 9 : 12 2:120

Median :103.0 Median :10.00 20 : 12 3:120

Mean :121.8 Mean :10.72 10 : 12 4:118

3rd Qu.:163.8 3rd Qu.:16.00 17 : 12

Max. :373.0 Max. :21.00 19 : 12

(Other):506

> table(ChickWeight$Chick)

18 16 15 13 9 20 10 8 17 19 4 6 11 3 1 12 2 5 14 7 24 30 22 23 27 28

2 7 8 12 12 12 12 11 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12

26 25 29 21 33 37 36 31 39 38 32 40 34 35 44 45 43 41 47 49 46 50 42 48

12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 12 12 12 12 12 12 12 12 12

> table(ChickWeight$Time)

0 2 4 6 8 10 12 14 16 18 20 21

50 50 49 49 49 49 49 48 47 47 46 45

> table(ChickWeight$Diet)

1 2 3 4

220 120 120 118

> table(ChickWeight$weight)

35 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56

1 8 5 20 15 4 1 1 2 1 8 13 6 8 4 5 3 5 4

57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 76

3 5 5 2 7 8 5 5 5 4 5 6 1 2 6 6 5 8 3

77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 95 96

3 3 4 2 2 3 2 6 6 2 4 2 5 3 1 2 4 1 5

97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115

1 7 1 4 4 3 7 1 1 4 3 3 2 1 2 4 1 2 4

116 117 118 119 120 122 123 124 125 126 127 128 129 130 131 133 134 135 136

2 2 1 1 4 3 5 4 4 2 2 3 1 1 2 1 3 4 2

137 138 139 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156

1 4 1 3 1 1 4 4 5 3 3 1 2 2 1 1 4 3 3

157 158 159 160 162 163 164 166 167 168 169 170 171 173 174 175 177 178 179

7 1 1 4 2 4 5 2 1 2 2 4 1 1 3 4 1 1 1

181 182 184 185 186 187 188 192 195 196 197 198 199 200 201 202 203 204 205

1 2 5 3 1 3 1 3 1 1 4 3 4 1 1 1 1 3 5

207 209 210 212 214 215 217 218 220 221 222 223 225 227 230 231 232 233 234

1 1 2 1 1 2 1 1 2 1 1 1 1 2 1 2 1 2 2

235 236 237 238 240 248 250 251 256 259 261 262 263 264 265 266 269 272 275

2 1 1 2 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1

279 280 281 287 288 290 291 294 295 303 305 307 309 318 321 322 327 331 332

1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

341 361 373

1 1 1

>   
(ii)Extract last 6 records of dataset

**programm:**

> tail(ChickWeight, 6)

**output:**

weight Time Chick Diet

573 155 12 50 4

574 175 14 50 4

575 205 16 50 4

576 234 18 50 4

577 264 20 50 4

578 264 21 50 4

>   
(iii) order the data frame, in ascending order by feature name  “weight”  grouped by feature “diet”  
(iv)Perform melting function based on “Chick","Time","Diet"   features as ID variables  
(v)Perform cast function to display the mean value of weight grouped by Diet  
  
  
3.(i)Get the Statistical  Summary of  “ChickWeight” dataset

**programm:**

> data("ChickWeight")

> summary(ChickWeight)

**output:**

weight Time Chick Diet

Min. : 35.0 Min. : 0.00 13 : 12 1:220

1st Qu.: 63.0 1st Qu.: 4.00 9 : 12 2:120

Median :103.0 Median :10.00 20 : 12 3:120

Mean :121.8 Mean :10.72 10 : 12 4:118

3rd Qu.:163.8 3rd Qu.:16.00 17 : 12

Max. :373.0 Max. :21.00 19 : 12

(Other):506   
 (ii)Create Box plot for “weight”  grouped by “Diet”  
 (iii)Create a Histogram for  “Weight” features  belong to Diet- 1 category

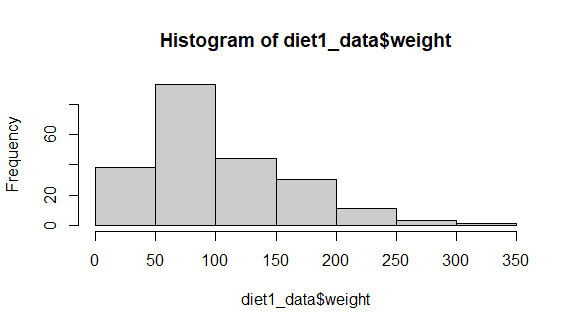
**programm:**

diet1\_data <- subset(ChickWeight, Diet == 1)

> hist(diet1\_data$weight)

>

**output:**



(iv) Create a Histogram for  “Weight” features  belong to Diet- 4 category

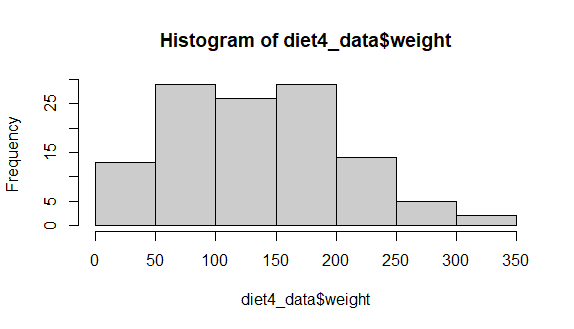
**programm:**

> diet4\_data <- subset(ChickWeight, Diet == 4)

> hist(diet4\_data$weight)

>

**output:**



(v) Create Scatter plot  for weight vs Time grouped by Diet  
**programm:**  
  
4.(i) Create multi regression model to find a weight of the chicken , by “Time” and “Diet” as as predictor variables

**programm:**

library(lmtest)

> model <- lm(weight ~ Time + Diet, data = ChickWeight)

>

> summary(model)

**output:**

Call:

lm(formula = weight ~ Time + Diet, data = ChickWeight)

Residuals:

Min 1Q Median 3Q Max

-136.851 -17.151 -2.595 15.033 141.816

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 10.9244 3.3607 3.251 0.00122 \*\*

Time 8.7505 0.2218 39.451 < 2e-16 \*\*\*

Diet2 16.1661 4.0858 3.957 8.56e-05 \*\*\*

Diet3 36.4994 4.0858 8.933 < 2e-16 \*\*\*

Diet4 30.2335 4.1075 7.361 6.39e-13 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 35.99 on 573 degrees of freedom

Multiple R-squared: 0.7453, Adjusted R-squared: 0.7435

F-statistic: 419.2 on 4 and 573 DF, p-value: < 2.2e-16  
(ii) Predict weight for Time=10 and Diet=1

**programm:**

pred\_data <- data.frame(Time = 10, Diet = 1)

>

> prediction <- predict(model, newdata = pred\_data)

> prediction

**output:**

1

1392.46

>   
(iii)Find the error in model for smae

**programm:**

residuals <- residuals(model)

>

> mse <- mean(residuals^2)

> mse

**output:**

[1] 1284.319