Write an informal report to Danielle about:

* your experience switching to R,
* the errors that you encountered in the task,
* the reasons behind them,
* and how you overcame them.
* You should also include discussion of the outcomes your model predicted.

**attributes(cars)**

#List your attributes within your data set. In our case we have three attributes:

$names

* name.of.car
* speed.of.car
* distance.of.car

$class

[1] "data.frame"

And it shows the names of the rows, I don’t know why the first 1 and on the second row the 34 are in brackets, but I think it is because it is the beginning of a new line, it could also mean the number of instances:

$row.names

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

[34] 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

**summary(cars)**

#Prints the min, max, mean, median, and quartiles of each attribute.

name.of.car speed.of.car distance.of.car

Dodge: 3 Min.: 4.0 Min.: 2.00

Honda: 3 1st Qu.:12.0 1st Qu.: 26.00

Jeep: 3 Median :15.0 Median: 36.00

KIA: 3 Mean :15.4 Mean : 42.98

Acura: 2 3rd Qu.:19.0 3rd Qu.: 56.00

Audi: 2 Max. :25.0 Max. :120.00

(Other):34

Other: 34 means there are 34 more cars in the list. Of two per brand or less.

**str(cars)**

#displays the structure of the dataset

**names(cars)**

#names your attributes, same as attributes)cars)

**cars$speed.of.car (cars$ColumnName)**

#Will print out the instances within that particular column in your data set.

[1] 4 4 7 7 8 9 10 10 10 11 11 12 12 12 12 13 13 13 13 14 14 14 14 15 15 15 16 16 17 17 17 18 18

[34] 18 18 19 19 19 20 20 20 20 20 22 23 24 24 24 24 25

**cars$name.of.car**

[1] Ford Jeep Honda KIA Toyota BMW Mercedes GM Hyundai

[10] Infiniti Land Rover Lexus Mazda Mitsubishi Nissan GMC Fiat Chrysler

[19] Dodge Acura Audi Chevrolet Buick Ford Jeep Honda KIA

[28] Toyota BMW Mercedes GM Hyundai Infiniti Land Rover Lexus Mazda

[37] Mitsubishi Nissan GMC Fiat Chrysler Dodge Acura Audi Chevrolet

[46] Buick Jeep Honda KIA Dodge

23 Levels: Acura Audi BMW Buick Chevrolet Chrysler Dodge Fiat Ford GM GMC Honda Hyundai ... Toyota

**cars$distance.of.car**

[1] 2 4 10 10 14 16 17 18 20 20 22 24 26 26 26 26 28 28 32 32 32 34 34 34 36

[26] 36 40 40 42 46 46 48 50 52 54 54 56 56 60 64 66 68 70 76 80 84 85 92 93 120

**How to convert categorical columns to numerical:**

Numerical =

Factor = nominal data, no ranking. Line of code: is.factor(NameofData$ColumnName) in our case: is.factor(cars$name.of.car)

Logical =

**> is.numeric(cars$name.of.car)**

**[1] FALSE**

**> is.numeric(cars$speed.of.car)**

**[1] TRUE**

**To change factor to numeric:**

> cars$name.of.car = as.numeric(cars$name.of.car)

> is.numeric(cars$name.of.car)

[1] TRUE

**> cars$name.of.car**

[1] 9 15 12 16 23 3 20 10 13 14 17 18 19 21 22 11 8 6

[19] 7 1 2 5 4 9 15 12 16 23 3 20 10 13 14 17 18 19

[37] 21 22 11 8 6 7 1 2 5 4 15 12 16 7

> summary(cars)

name.of.car speed.of.car distance.of.car

Min. : 1.00 Min. : 4.0 Min. : 2.00

1st Qu.: 7.00 1st Qu.:12.0 1st Qu.: 26.00

Median :12.00 Median :15.0 Median : 36.00

Mean :12.04 Mean :15.4 Mean : 42.98

3rd Qu.:17.00 3rd Qu.:19.0 3rd Qu.: 56.00

Max. :23.00 Max. :25.0 Max. :120.00

**To change factor to numeric:**

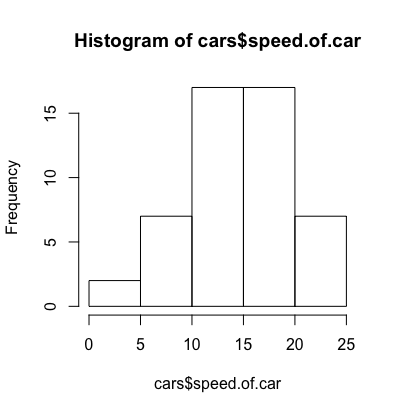
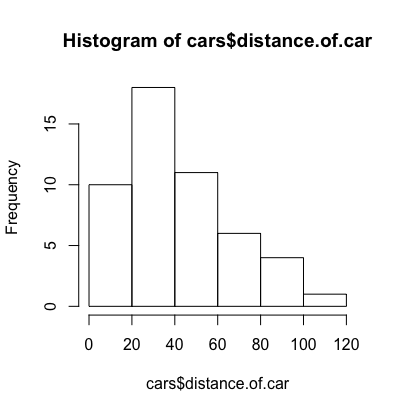
> cars$name.of.car = as.numeric(cars$name.of.car)

> is.numeric(cars$name.of.car)

[1] TRUE

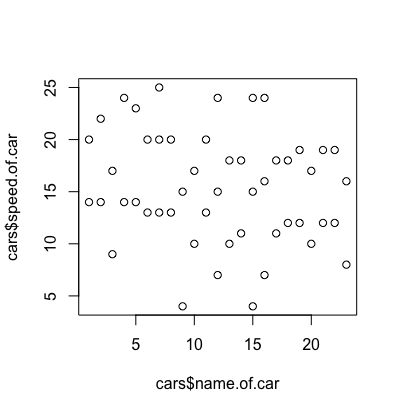
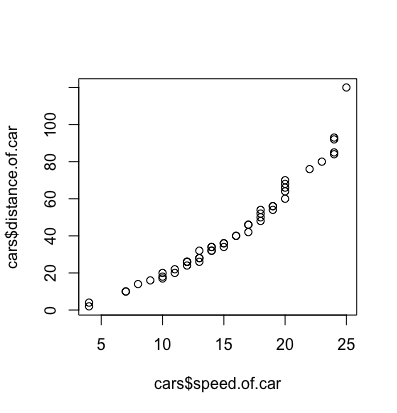
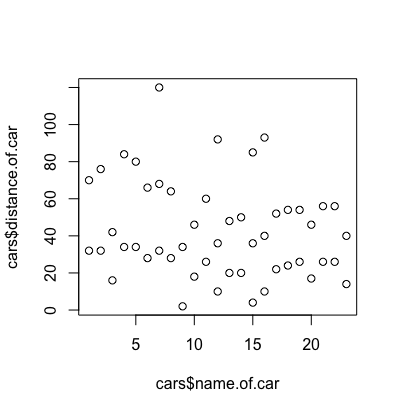
**Histogram Plot**

hist(DatasetName$ColumnName)

****

**Scatter (Box) Plot**

plot(DatasetName$ColumnName,DatasetName$ColumnName)

****

**Normal Quantile Plot**- is a way to see if your data is normally distributed.

qqnorm(DatasetName$ColumnName)

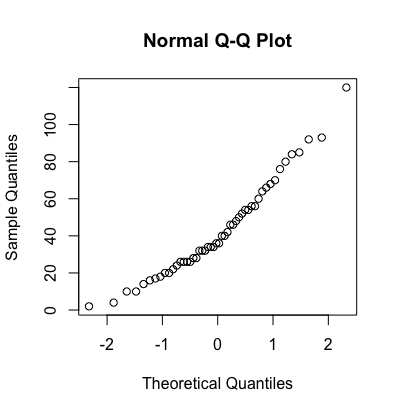
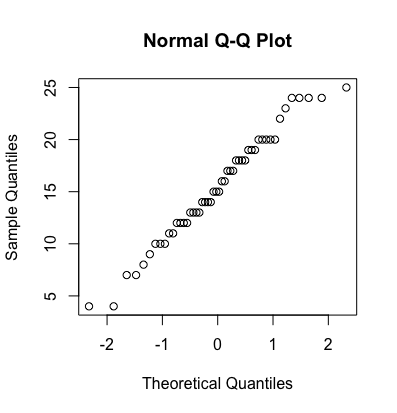


Figure 1 Distance & Figure 2 Speed

In a QQ plot we compare the quantiles of our empirical data with the ideal. Do the points fall in a straight line? Both QQ plots show almost a straight line. On the left QQ plot we see a slight dip in the middle and the right plot shows a few points outside the line on the most extreme sides. Theoretical quantiles are the ideal quantiles, sample quantiles are the quantiles of our data. However, it is not too much far off to concern us, therefore we consider it normally distributed.

Skewed – wave of the line, extreme values to left or right?

Kurtosis – size of the curve, slim and high or fat and low

Boxplots are also tools to see if normally distributed, if symmetrical then normal. Q1 and Q3 similar in distance to mean, whiskers same length.

**CLEANING/PREPROCESSING YOUR DATA**

A vector is a sequence of the same **data type.**

**Here are the data types that you will encounter when working in R:**

* **Numeric**- Numbers with decimals. (Ex: 1.0, 10.5, 4.5, etc.)
* **Integer Data**- Whole numbers (Ex: 11, 45, 78, etc.)
* **Factor Data**- Categorical data (Ex: Red, Blue, Green, Purple, etc.)
* **Ordinal Data**- Ordered data (Ex: educational levels, temperature, etc.)
* **Character Data**- String values, which are characters (words) with quotes around them. (Ex: "Red", "Blue", "Green", "Purple", etc.)
* **Logical**- TRUE or TRUE (Always capitalize TRUE or FALSE)

**TIP:**

Some common data transitions are numeric to factor, character to factor, integer to numeric, etc.

In our cars dataset:

* Name of the car is a factor
* Speed of the car is numeric
* Distance of the car is numeric

We need to change name of the car back to factor. And I am going to rename my attributes to NAME, DISTANCE and SPEED.

Change data types:

DatasetName$ColumnName<-as.typeofdata(DatasetName$ColumnName)

Rename attributes/columns:

names(DatasetName)<-c("ColumnName","ColumnName","ColumnName")

> names(cars)<-c("name","speed","distance")

> cars$name<-as.factor(cars$name)

> is.factor(cars$name)

[1] TRUE

**Missing values**

2 options: removal or replacing missing values with the mean.

summary(DatasetName) #Will count how many NA’s you have.

is.na(DatasetName) #Will show your NA’s through logical data. (TRUE if it’s missing, FALSE if it’s not.)

**Remove any observations containing missing data. (**If the missing data is less than 10% of the total data and only after comparing the min/max of all the features both with and without the missing data.)

na.omit(**DatasetName**$**ColumnName**#Drops any rows with missing values and omits them forever.

na.exclude(**DatasetName**$**ColumnName**)#Drops any rows with missing values, but keeps track of where they were.

**Replace the missing values with the mean**, which is common technique, but something to use with care with as it can skew the data.

DatasetName$ColumnName[is.na(DatasetName$ColumnName)]<-mean(DatasetName$ColumnName,na.rm = TRUE)

CREATING TESTING AND TRAINING SETS

Note: random seed number is pseudo random. Product of an algorithm. Setting the seed is useful if you want to set the same random numbers repeatedly.

**How to begin? In order to create your training and testing sets, you need to use the set.seed() function.**The seed is a number that you choose for a starting point used to create a sequence of random numbers. It is also helpful for others who want to recreate your same results. Here is the function:

**TIP:**

A common set.seed number is 123. To use the same set of random numbers, you’ll want to use the same seed number throughout your modeling process.

set.seed(###)

Split data 70/30 or 80/20

**These two lines calculate the sizes of each set but do not create the sets:**

trainSize<-round(nrow(DatasetName)\*0.7)

testSize<-nrow(DatasetName)-trainSize

**How do you create the training and test sets?** It’s now time for you to create the training and test sets. We also want these sets to be in a randomized order, which will create the most optimal model.

**To perform this, you need to run these three lines of code. Type in this code into R Script or Console:**

training\_indices<-sample(seq\_len(nrow(DatasetName)),size =trainSize)

trainSet<-DatasetName[training\_indices,]

testSet<-DatasetName[-training\_indices,]

* Preprocessing in R
* Data types in R
* How to change data types in R
* How to change names of columns in R?
* Changing names of columns in R
* Missing values in R
* How to handle missing values in R?
* How do you sample your data in R?
* How to test and train in R?
* How to split your data 70/30 in R?
* What is set.seed?
* What is a linear regression model?

**Linear regression in R**

Always check the residual plots.

Use linear model “lm” to fit a regression line through these

> model1<-lm(distance~ speed, trainSet)

> summary(model1)

abline(model1, col="red")

Call:

lm(formula = distance ~ speed, data = trainSet)

Residuals:

Min 1Q Median 3Q Max

-6.974 -4.307 -1.640 3.968 12.023

Coefficients:

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

(Intercept) -25.6964 2.8442 -9.035 1.93e-10

speed 4.4447 0.1754 25.342 < 2e-16

(Intercept) \*\*\*

speed \*\*\*

---

Signif. codes:

0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

**Y=4,447X-25,694**

**Residual standard error: 5.534 on 33 degrees of freedom**

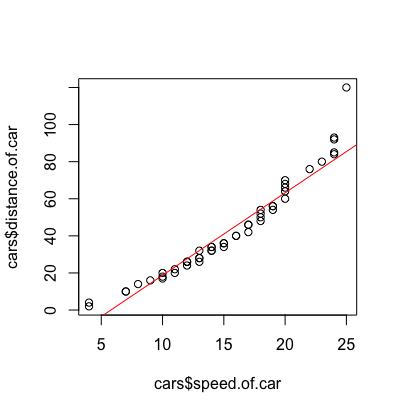
**Multiple R-squared: 0.9511, Adjusted R-squared: 0.9496**

**F-statistic: 642.2 on 1 and 33 DF, p-value: < 2.2e-16**

df = degrees of freedom

*H0 rejected by P<0.05*

*H0 not rejected by P>0.05*

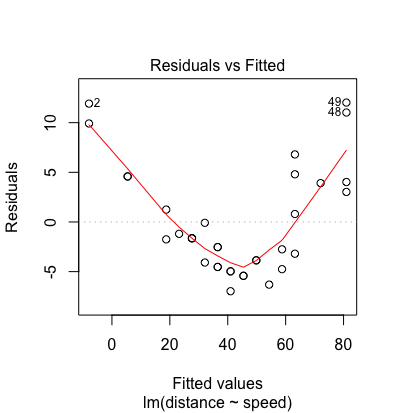


My p-value < .00000000000000022

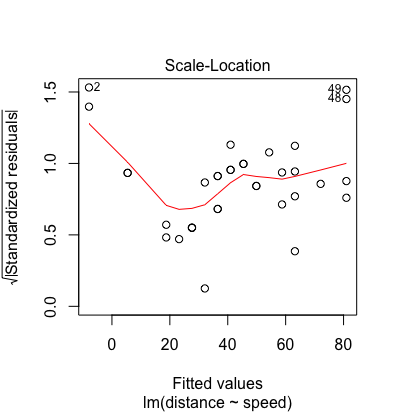
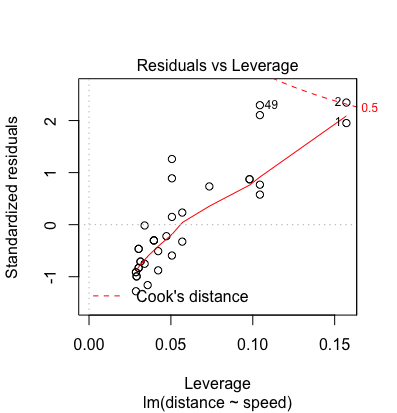
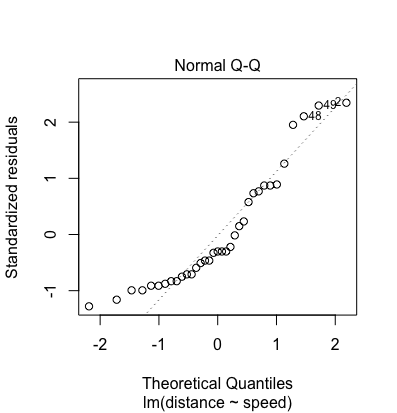
P < 0.05 reject H0

P> 0.05

Meaning of intercept: value y by x=0



>plot(model 1) # to investigate and look for residuals.



**Predictions**

> PredictionsName

5 6 8 10 12

9.86144 14.30616 18.75089 23.19561 27.64034

16 17 29 33 34

32.08506 32.08506 49.86397 54.30869 54.30869

35 37 41 45 50

54.30869 58.75342 63.19814 76.53232 85.42177

* Linear Regression Model in R or lm() function
* Multiple R-squared in R
* p-value in R
* Adjusted R-squared in R
* F-statistic in R
* How to interpret linear regression model's results?
* What is the predict function in R?
* How to predict in R?

**SetSeed (123)**

Call:

lm(formula = Petal.Length ~ Petal.Width, data = trainSet)

Residuals:

Min 1Q Median 3Q Max

-1.23056 -0.35734 -0.02579 0.40251 1.06759

Coefficients:

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

(Intercept) 1.0795 0.2307 4.68 6.65e-05 \*\*\*

Petal.Width 2.2831 0.1640 13.92 4.18e-14 \*\*\*

---

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

Residual standard error: 0.5785 on 28 degrees of freedom

Multiple R-squared: 0.8737, Adjusted R-squared: 0.8692

F-statistic: 193.7 on 1 and 28 DF, p-value: 4.185e-14

**Prediction SetSeed (123)**

1 2 3 4 5 8 9 10 11

1.536108 1.536108 1.536108 1.536108 1.536108 1.536108 1.536108 1.307801 1.536108

12 13 15 16 17 19 20 21 22

1.536108 1.307801 1.536108 1.992722 1.992722 1.764415 1.764415 1.536108 1.992722

23 24 25 26 27 28 29 30 31

1.536108 2.221029 1.536108 1.536108 1.992722 1.536108 1.536108 1.536108 1.536108

32 34 35 37 38 39 40 41 42

1.992722 1.536108 1.536108 1.536108 1.307801 1.536108 1.536108 1.764415 1.764415

43 45 46 47 48 49 50 51 52

1.536108 1.992722 1.764415 1.536108 1.536108 1.536108 1.536108 4.275794 4.504101

53 54 55 56 57 58 59 60 62

4.504101 4.047487 4.504101 4.047487 4.732409 3.362565 4.047487 4.275794 4.504101

63 66 67 69 70 71 72 73 75

3.362565 4.275794 4.504101 4.504101 3.590873 5.189023 4.047487 4.504101 4.047487

76 78 80 81 84 85 86 87 88

4.275794 4.960716 3.362565 3.590873 4.732409 4.504101 4.732409 4.504101 4.047487

91 92 93 95 96 97 98 99 100

3.819180 4.275794 3.819180 4.047487 3.819180 4.047487 4.047487 3.590873 4.047487

101 102 103 104 105 106 107 108 109

6.787173 5.417330 5.873945 5.189023 6.102252 5.873945 4.960716 5.189023 5.189023

110 111 112 113 114 115 117 119 120

6.787173 5.645637 5.417330 5.873945 5.645637 6.558866 5.189023 6.330559 4.504101

121 123 124 125 129 131 132 133 135

6.330559 5.645637 5.189023 5.873945 5.873945 5.417330 5.645637 6.102252 4.275794

136 137 139 140 141 143 144 145 146

6.330559 6.558866 5.189023 5.873945 6.558866 5.417330 6.330559 6.787173 6.330559

147 148 149

5.417330 5.645637 6.330559

**SetSeed (405)**

Call:

lm(formula = Petal.Length ~ Petal.Width, data = trainSet)

Residuals:

Min 1Q Median 3Q Max

-1.08464 -0.36536 0.01438 0.30866 1.01471

Coefficients:

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

(Intercept) 0.9677 0.1816 5.33 1.13e-05 \*\*\*

Petal.Width 2.3987 0.1263 18.99 < 2e-16 \*\*\*

---

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

Residual standard error: 0.4783 on 28 degrees of freedom

Multiple R-squared: 0.9279, Adjusted R-squared: 0.9254

F-statistic: 360.5 on 1 and 28 DF, p-value: < 2.2e-16

**Prediction SetSeed (405)**

1 2 3 4 5 8 9 10 11

1.536108 1.536108 1.536108 1.536108 1.536108 1.536108 1.536108 1.307801 1.536108

12 13 15 16 17 19 20 21 22

1.536108 1.307801 1.536108 1.992722 1.992722 1.764415 1.764415 1.536108 1.992722

23 24 25 26 27 28 29 30 31

1.536108 2.221029 1.536108 1.536108 1.992722 1.536108 1.536108 1.536108 1.536108

32 34 35 37 38 39 40 41 42

1.992722 1.536108 1.536108 1.536108 1.307801 1.536108 1.536108 1.764415 1.764415

43 45 46 47 48 49 50 51 52

1.536108 1.992722 1.764415 1.536108 1.536108 1.536108 1.536108 4.275794 4.504101

53 54 55 56 57 58 59 60 62

4.504101 4.047487 4.504101 4.047487 4.732409 3.362565 4.047487 4.275794 4.504101

63 66 67 69 70 71 72 73 75

3.362565 4.275794 4.504101 4.504101 3.590873 5.189023 4.047487 4.504101 4.047487

76 78 80 81 84 85 86 87 88

4.275794 4.960716 3.362565 3.590873 4.732409 4.504101 4.732409 4.504101 4.047487

91 92 93 95 96 97 98 99 100

3.819180 4.275794 3.819180 4.047487 3.819180 4.047487 4.047487 3.590873 4.047487

101 102 103 104 105 106 107 108 109

6.787173 5.417330 5.873945 5.189023 6.102252 5.873945 4.960716 5.189023 5.189023

110 111 112 113 114 115 117 119 120

6.787173 5.645637 5.417330 5.873945 5.645637 6.558866 5.189023 6.330559 4.504101

121 123 124 125 129 131 132 133 135

6.330559 5.645637 5.189023 5.873945 5.873945 5.417330 5.645637 6.102252 4.275794

136 137 139 140 141 143 144 145 146

6.330559 6.558866 5.189023 5.873945 6.558866 5.417330 6.330559 6.787173 6.330559

147 148 149

5.417330 5.645637 6.330559

install.packages("readr")

library("readr")

IrisDataset <- read.csv("iris.csv")

attributes(IrisDataset)

summary(IrisDataset)

str(IrisDataset)

names(IrisDataset)

#turn species into numeric

IrisDataset$Species = as.numeric(IrisDataset$Species)

#histograms

hist(IrisDataset$Species)

hist(IrisDataset$Sepal.Length)

hist(IrisDataset$Sepal.Width)

hist(IrisDataset$Petal.Length)

hist(IrisDataset$Petal.Width)

#plots (first x then y)

plot(IrisDataset$Petal.Width, IrisDataset$Petal.Length)

#QQ qqnorm(NameDateSet$ColumnName)

qqnorm(IrisDataset$Petal.Length)

qqnorm(IrisDataset$Petal.Width)

#set seed random sample

set.seed(123)

trainSize <- round(nrow(IrisDataset) \* 0.2)

testSize <- nrow(IrisDataset) - trainSize

trainSize

testSize

#set training indices

training\_indices<-sample(seq\_len(nrow(IrisDataset)),size =trainSize)

trainSet <- IrisDataset[training\_indices, ]

testSet <- IrisDataset[-training\_indices, ]

#train linear regression model, dependent variable first, then independent variable, use trainset!

#NameModel <- lm(dependentVariable~independentVariable, trainSet)

LinearModel <- lm(Petal.Length~Petal.Width,trainSet)

summary(LinearModel)

#Look at the residuals

plot(LinearModel)

#Make predictions on the testSet

prediction<-predict(LinearModel,testSet)

prediction

plot(prediction)

write.table(prediction)

|  |  |
| --- | --- |
| X= petal width | Y = petal length |
| 1 | 1.53610776035384 |
| 2 | 1.53610776035384 |
| 3 | 1.53610776035384 |
| 4 | 1.53610776035384 |
| 5 | 1.53610776035384 |
| 8 | 1.53610776035384 |
| 9 | 1.53610776035384 |
| 10 | 1.30780056292722 |
| 11 | 1.53610776035384 |
| 12 | 1.53610776035384 |
| 13 | 1.30780056292722 |
| 15 | 1.53610776035384 |
| 16 | 1.99272215520708 |
| 17 | 1.99272215520708 |
| 19 | 1.76441495778046 |
| 20 | 1.76441495778046 |
| 21 | 1.53610776035384 |
| 22 | 1.99272215520708 |
| 23 | 1.53610776035384 |
| 24 | 2.22102935263369 |
| 25 | 1.53610776035384 |
| 26 | 1.53610776035384 |
| 27 | 1.99272215520708 |
| 28 | 1.53610776035384 |
| 29 | 1.53610776035384 |
| 30 | 1.53610776035384 |
| 31 | 1.53610776035384 |
| 32 | 1.99272215520708 |
| 34 | 1.53610776035384 |
| 35 | 1.53610776035384 |
| 37 | 1.53610776035384 |
| 38 | 1.30780056292722 |
| 39 | 1.53610776035384 |
| 40 | 1.53610776035384 |
| 41 | 1.76441495778046 |
| 42 | 1.76441495778046 |
| 43 | 1.53610776035384 |
| 45 | 1.99272215520708 |
| 46 | 1.76441495778046 |
| 47 | 1.53610776035384 |
| 48 | 1.53610776035384 |
| 49 | 1.53610776035384 |
| 50 | 1.53610776035384 |
| 51 | 4.27579412947326 |
| 52 | 4.50410132689988 |
| 53 | 4.50410132689988 |
| 54 | 4.04748693204664 |
| 55 | 4.50410132689988 |
| 56 | 4.04748693204664 |
| 57 | 4.7324085243265 |
| 58 | 3.36256533976679 |
| 59 | 4.04748693204664 |
| 60 | 4.27579412947326 |
| 62 | 4.50410132689988 |
| 63 | 3.36256533976679 |
| 66 | 4.27579412947326 |
| 67 | 4.50410132689988 |
| 69 | 4.50410132689988 |
| 70 | 3.59087253719341 |
| 71 | 5.18902291917973 |
| 72 | 4.04748693204664 |
| 73 | 4.50410132689988 |
| 75 | 4.04748693204664 |
| 76 | 4.27579412947326 |
| 78 | 4.96071572175312 |
| 80 | 3.36256533976679 |
| 81 | 3.59087253719341 |
| 84 | 4.7324085243265 |
| 85 | 4.50410132689988 |
| 86 | 4.7324085243265 |
| 87 | 4.50410132689988 |
| 88 | 4.04748693204664 |
| 91 | 3.81917973462002 |
| 92 | 4.27579412947326 |
| 93 | 3.81917973462002 |
| 95 | 4.04748693204664 |
| 96 | 3.81917973462002 |
| 97 | 4.04748693204664 |
| 98 | 4.04748693204664 |
| 99 | 3.59087253719341 |
| 100 | 4.04748693204664 |
| 101 | 6.78717330116607 |
| 102 | 5.41733011660635 |
| 103 | 5.87394451145959 |
| 104 | 5.18902291917973 |
| 105 | 6.10225170888621 |
| 106 | 5.87394451145959 |
| 107 | 4.96071572175312 |
| 108 | 5.18902291917973 |
| 109 | 5.18902291917973 |
| 110 | 6.78717330116607 |
| 111 | 5.64563731403297 |
| 112 | 5.41733011660635 |
| 113 | 5.87394451145959 |
| 114 | 5.64563731403297 |
| 115 | 6.55886610373945 |
| 117 | 5.18902291917973 |
| 119 | 6.33055890631283 |
| 120 | 4.50410132689988 |
| 121 | 6.33055890631283 |
| 123 | 5.64563731403297 |
| 124 | 5.18902291917973 |
| 125 | 5.87394451145959 |
| 129 | 5.87394451145959 |
| 131 | 5.41733011660635 |
| 132 | 5.64563731403297 |
| 133 | 6.10225170888621 |
| 135 | 4.27579412947326 |
| 136 | 6.33055890631283 |
| 137 | 6.55886610373945 |
| 139 | 5.18902291917973 |
| 140 | 5.87394451145959 |
| 141 | 6.55886610373945 |
| 143 | 5.41733011660635 |
| 144 | 6.33055890631283 |
| 145 | 6.78717330116607 |
| 146 | 6.33055890631283 |
| 147 | 5.41733011660635 |
| 148 | 5.64563731403297 |
| 149 | 6.33055890631283 |