Regression Type	Formula	Description
Linear Regression	$y = \beta_0 + \beta_1 x_1 + \epsilon$	where: y is the dependent variable. x is the independent variable. β_0 is the y-intercept. β_1 is the slope of the line. ϵ is the error term (residual)
Multiple Linear Regression	$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + .. \beta_n x_n + \epsilon$	where: y is the dependent variable. $x_1, x_2, x_3, \ldots . x_n$ are the independent variables. $\beta_1, \beta_2, \beta_3, \ldots . \beta_n$ are the coefficients. ϵ is the error term (residual)
Logistic Regression	$P(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots\ldots \beta_n x_n)}}$	where: y is the dependent variable. $x1, x2, x3, \ldots . xn$ are the independent variables. $\beta 1, \beta 2, \beta 3, \ldots . \beta n$ are the coefficients. e Is Euler's Coeffiecient,

Aspect	Description
Algorithm Name	DGIM (Datar-Gionis-Indyk-Motwani) Algorithm
Purpose	Approximate the number of 1s in the last N bits of a binary stream
Application	Processing large data streams with limited memory
Key Idea	Maintain a summary of the stream using buckets, each representing a group of 1s
Bucket Properties	• Stores timestamp of most recent 1 • Sizes are powers of 2 (1, 2, 4, 8, etc.) • At most two buckets of each size
Process	1. Create a bucket of size 1 for new 1s 2. Merge oldest two buckets if more than two of a size exist 3. Remove buckets outside the window
Querying	1. Sum sizes of buckets fully within the window 2. Add fraction of oldest bucket partially in window
Accuracy	Guarantees approximation within 50% of true count
Space Complexity	O(log N) buckets, where N is the window size

MongoDB Commands:

Command	Description	Syntax Example
Database Operations		
show dbs	Lists all databases.	show dbs
use <database>	Switches to the specified database. If it doesn't exist, it will be created upon inserting data.	use myDatabase
db	Displays the current database.	db
db.dropDatabase()	Deletes the current database.	db.dropDatabase()
Collection Operations		
show collections	Lists all collections in the current database.	show collections
db.createCollection()	Creates a new collection.	db.createCollection("myCollection")
db.<collection>.drop()	Drops the specified collection.	db.myCollection.drop()
Insert Operations		
db.<collection>.insertOne()	Inserts a single document into the collection.	db.myCollection.insertOne({ name: "John", age: 30 })
db.<collection>.insertMany()	Inserts multiple documents into the collection.	db.myCollection.insertMany([{ name: "Jane" }, { name: "Doe" }])
Update Operations		
db.<collection>.updateOne()	Updates a single document that matches the filter criteria.	db.myCollection.updateOne({ name: "John" }, { $set: { age: 31 } })
db.<collection>.updateMany()	Updates all documents that match the filter criteria.	db.myCollection.updateMany({ name: "John" }, { $set: { age: 32 } })
db.<collection>.replaceOne()	Replaces a single document that matches the filter criteria.	db.myCollection.replaceOne({ name: "John" }, { name: "John Doe", age: 33 })
Delete Operations		
db.<collection>.deleteOne()	Deletes a single document that matches the filter criteria.	db.myCollection.deleteOne({ name: "John" })
db.<collection>.deleteMany()	Deletes all documents that match the filter criteria.	db.myCollection.deleteMany({ name: "John" })
Query Operations		
db.<collection>.find()	Finds all documents in a collection that match the query criteria.	db.myCollection.find({ age: { $gt: 25 } })
db.<collection>.findOne()	Finds the first document in a collection that matches the query criteria.	db.myCollection.findOne({ name: "Jane" })
Indexing		
db.<collection>.createIndex()	Creates an index on a field or fields.	db.myCollection.createIndex({ age: 1 })
db.<collection>.getIndexes()	Returns a list of all indexes for a collection.	db.myCollection.getIndexes()
db.<collection>.dropIndex()	Drops a specified index from a collection.	db.myCollection.dropIndex("age_1")

Aggregation Operations		
db.<collection>.aggregate()	Performs aggregation operations like grouping, sorting, and averaging.	db.myCollection.aggregate([{ $match: { age: { $gt: 25 } } }, { $group: { _id: "$age", total: { $sum: 1 } } }])
Miscellaneous Commands		
db.stats()	Provides statistics about the current database.	db.stats()
db.<collection>.stats()	Provides statistics about a specific collection.	db.myCollection.stats()

Part B CRUD Commands Implementation

- **User based commands: getUsers(), createUser()**

```
test> db.getUsers()
{ users: [], ok: 1 }
test> use admin
switched to db admin
admin> db.createUser({
...   user: "nexus",
...   pwd: "12345",
...   roles: [{ role: "readWrite", db: "databaseName" }]
... })
{ ok: 1 }
admin> db.getUsers()
{
  users: [
    {
      _id: 'admin.nexus',
      userId: UUID('4c7fe888-60c6-4885-af42-45e00c0e16dc'),
      user: 'nexus',
      db: 'admin',
      roles: [ { role: 'readWrite', db: 'databaseName' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    }
  ],
  ok: 1
}
```

- **Database Commands: Create database, show databases**

```
admin> use  BigDB
switched to db BigDB
BigDB>
```

```
BigDB> show databases
BigDB    72.00 KiB
admin   132.00 KiB
config   96.00 KiB
local    72.00 KiB
BigDB>
```

- **Insert Commands: insertOne(), Insert Many**

```
BigDB> db.books.insertOne({ title: "The Great Gatsby", author: "F. Scott Fitzgerald",
 year: 1925 })
{
  acknowledged: true,
  insertedId: ObjectId('66b9b1567dfb0fafb3838729')
}
BigDB> db.books.insertMany([
...   { title: "To Kill a Mockingbird", author: "Harper Lee", year: 1960 },
...   { title: "1984", author: "George Orwell", year: 1949 },
...   { title: "Moby-Dick", author: "Herman Melville", year: 1851 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66b9b15a7dfb0fafb383872a'),
    '1': ObjectId('66b9b15a7dfb0fafb383872b'),
    '2': ObjectId('66b9b15a7dfb0fafb383872c')
  }
}
```

```
BigDB> db.books.find().pretty()
[
  {
    _id: ObjectId('66b9b1567dfb0fafb3838729'),
    title: 'The Great Gatsby',
    author: 'F. Scott Fitzgerald',
    year: 1925
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872a'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1960
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872b'),
    title: '1984',
    author: 'George Orwell',
    year: 1949
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872c'),
    title: 'Moby-Dick',
    author: 'Herman Melville',
    year: 1851
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872d'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1960
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872e'),
    title: '1984',
    author: 'George Orwell',
    year: 1949
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872f'),
    title: 'Moby-Dick',
    author: 'Herman Melville',
    year: 1851
  }
]
```

- Find

- Find with Condition:

```
BigDB> db.books.find({ author: "George Orwell" }).pretty()
[
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872b'),
    title: '1984',
    author: 'George Orwell',
    year: 1949
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872e'),
    title: '1984',
    author: 'George Orwell',
    year: 1949
  }
]
BigDB> db.books.find({ year: { $gt: 1950 } }).pretty()
[
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872a'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1960
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872d'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1960
  }
]
BigDB> db.books.updateOne(
...    { title: "1984" },           // Query filter
...    { $set: { year: 1950 } }     // Update operation
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
BigDB> db.books.updateMany(
...    { author: "Harper Lee" },    // Query filter
...    { $set: { year: 1961 } }     // Update operation
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
BigDB>
```

- Find with limiting output

```
BigDB> db.books.find().limit(5).pretty()
[
  {
    _id: ObjectId('66b9b1567dfb0fafb3838729'),
    title: 'The Great Gatsby',
    author: 'F. Scott Fitzgerald',
    year: 1925
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872a'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1961
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872b'),
    title: '1984',
    author: 'George Orwell',
    year: 1950
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872c'),
    title: 'Moby-Dick',
    author: 'Herman Melville',
    year: 1851
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872d'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1961
  }
]
BigDB>
```

- Update Commands: updateOne(), updateMany()

```
BigDB> db.books.updateOne(
...   { title: '1984', author: 'George Orwell', year: 1950 },
...   { $set: { year: 1949 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
BigDB> db.books.updateMany(
...   { title: '1984' },
...   { $set: { year: 1950 } }
... )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

```
BigDB> db.books.find().pretty()
[
  {
    _id: ObjectId('66b9b1567dfb0fafb3838729'),
    title: 'The Great Gatsby',
    author: 'F. Scott Fitzgerald',
    year: 1925
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872a'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1961
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872b'),
    title: '1984',
    author: 'George Orwell',
    year: 1950
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872d'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1961
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872e'),
    title: '1984',
    author: 'George Orwell',
    year: 1950
  }
]
```

- Delete Commands: deleteOne(), deleteMany()

```
BigDB> db.books.deleteOne({ title: "Moby-Dick" })
{ acknowledged: true, deletedCount: 1 }
BigDB> db.books.deleteMany({ year: { $lt: 1900 } })
{ acknowledged: true, deletedCount: 0 }
BigDB> db.books.find().pretty()
[
  {
    _id: ObjectId('66b9b1567dfb0fafb3838729'),
    title: 'The Great Gatsby',
    author: 'F. Scott Fitzgerald',
    year: 1925
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872a'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1961
  },
  {
    _id: ObjectId('66b9b15a7dfb0fafb383872b'),
    title: '1984',
    author: 'George Orwell',
    year: 1950
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872d'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1961
  },
  {
    _id: ObjectId('66b9b15f7dfb0fafb383872e'),
    title: '1984',
    author: 'George Orwell',
    year: 1949
  }
]
```

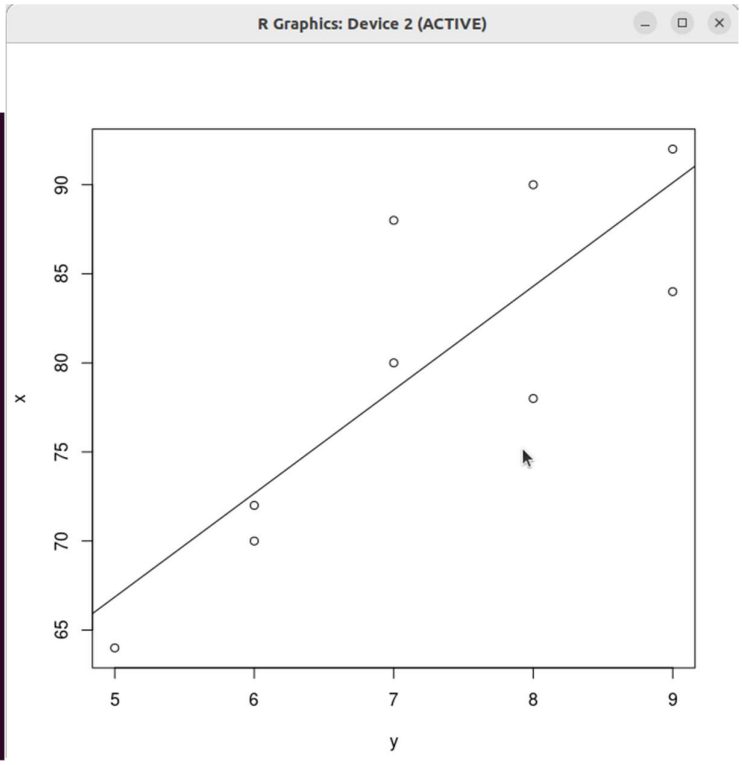
LINEAR REGRESSION →

```
> #Linear Regression
> x<-c(90,70,80,84,78,92,88,72,64)
> y<-c(8,6,7,9,8,9,7,6,5)
> relation<-lm(y~x)
> print(relation
+ )

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     -2.430        0.121

> a<-data.frame(x=94)
> result<-predict(relation,a)
> print(result)
       1
8.942925
>
> plot(y,x,abline(lm(x~y)))
>
```



STEP 01 Open R compiler and Load
#Get working Directory:
getwd()

#Download and load the Dataset:
download.file("https://cmu-lib.github.io/os-workshops/reproducible-research/data/interviews_plotting.csv","interviews_plotting.csv", mode = "wb")

#List all Files in Working Directory:
list.files()
# Load the dataset
data <- read.csv("interviews_plotting.csv")

# View the first few rows of the dataset
View(data)