Exercise III - R

Part 1 – Regression Analysis in R

CEE412/CET522

TRANSPORTATION DATA MANAGEMENT AND VISUALIZATION

WINTER 2020

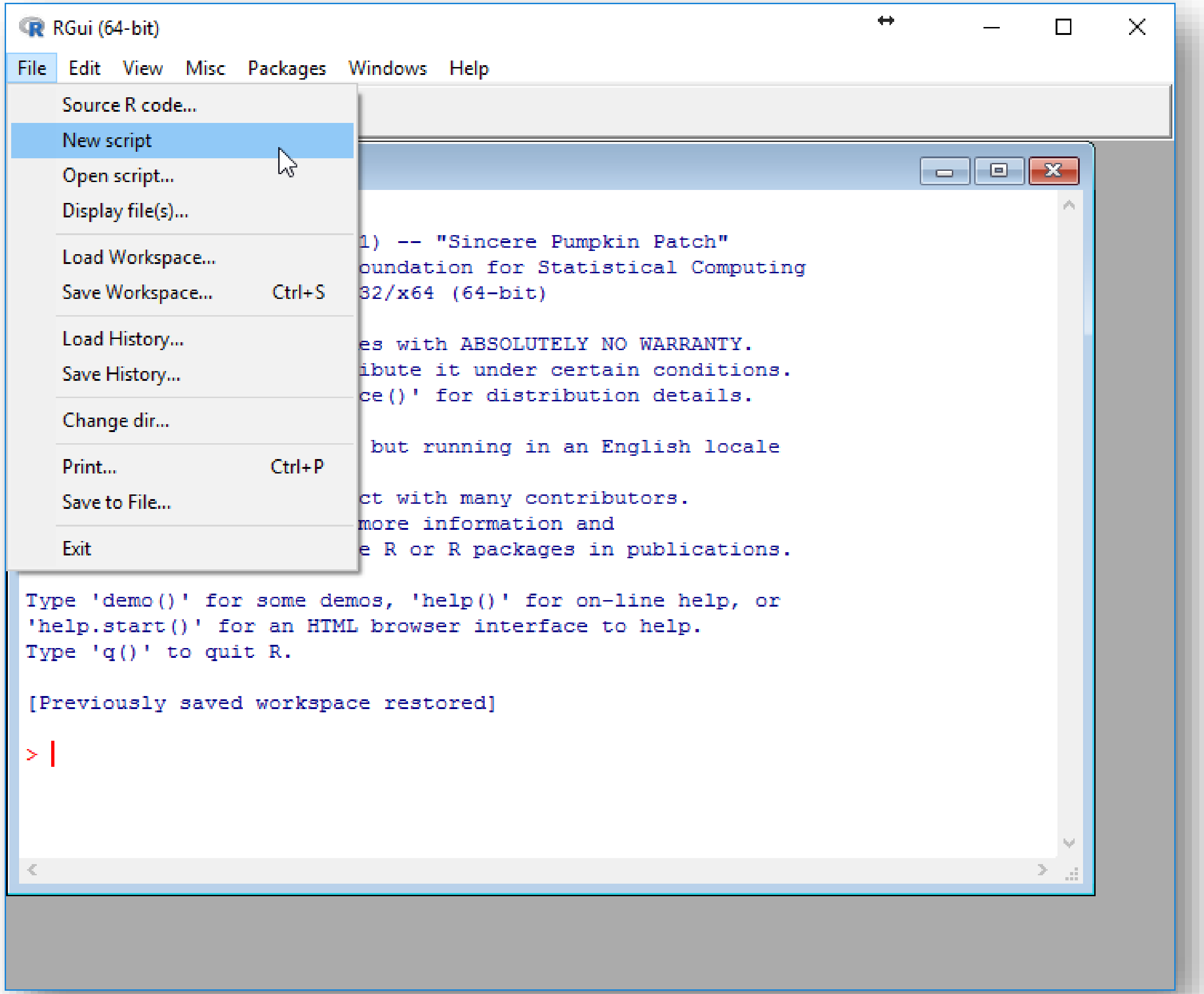
Getting Started

The objective of this exercise is to learn how to import data from SQL to R, and use R to model accident data.

We will start by some basic operations in R and then creating a data connection with RODBC to get accident data from SQL server.

Step 1: Load R

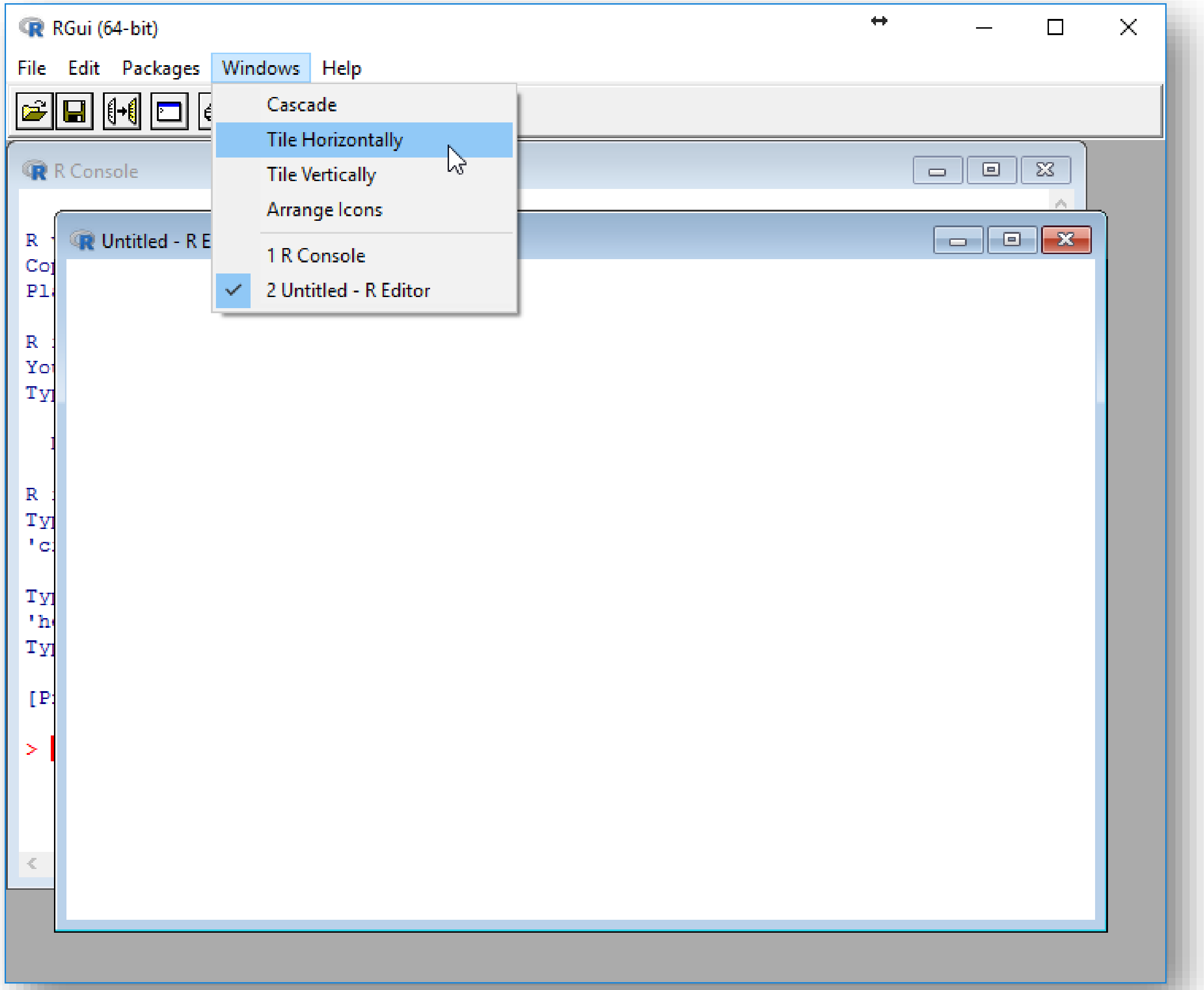
- Click **Start → R → R x-64** (or search "RGUI" in Cortana).
- After starting R, click **File → New script** to create a new script for editing R code as shown:



Step 2: Format R Environment

- You can manually change your window size by dragging or maximizing.

- Or click **Window** and pick a window layout as shown:



Step 3: Get Started in R

- You can skip this step if you are familiar with R.
- Type your code in to R Editor Window to practice some basic functions.
- Hit Ctrl+R on keyboard or click  to execute your current line or a selection of code.
- Try some codes as shown:

```
# create a numeric value (this is a comment)
a = 5
# create a vector, the rep() function here will replicate the value 2 for ten times
b = rep(2, 10)
# create a sequence of integers from 1 to 100
c = 1:10
# you can do simple arithmetic calculation between numeric values and vectors
d = a*b*c
# show your result in the Console Window
d
# select a subset of variables in your vector
d[3:5]
```

Step 4: Simple Linear Regression

- You can skip this if you are familiar with R.
- Create some vectors as shown:

```
# create a vector that takes 100 samples with replacement from 1 to 100
x1 = sample(1:100, 100, replace = TRUE)
# you can use "?" or help() to find the help document for a function
?sample
help(sample)
# generate 100 random observations from a normal distribution (mean=20, variance=5)
x2 = rnorm(100, 20, 5)
# generate 100 random observations from a Poisson distribution (lambda=10)
x3 = rpois(100, 10)

# create a vector y as the linear combination of previous three vectors,  and plus some random error
y = 0.8*x1 - 3*x2 + 4.5*x3 + rnorm(100, 0, 5)
```

Step 4: Simple Linear Regression (cont.)

- Try some operations on the dataframe as shown:

```
# combine vectors to create a dataframe using the data.frame() function.
# a period "." in R has no special meaning,
# and you can use it to name your objects in the same way as other characters
data.sample = data.frame(y, x1, x2, x3)

# take a look at the first 10 rows in the dataframe
data.sample[1:10,]

# use the dollar sign "$" to reference a column in the dataframe
data.sample$y
```

Step 4: Simple Linear Regression (cont.)

- Create some simple plots:

```
# create some plots to look at relationships between columns
plot(x1, y)
# you can also refer to dataframe columns to create a plot
plot(data.sample$x2, data.sample$y)
```

- Run a simple linear regression using the lm() function.
- Here, the regression equation is: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$

```
# simple linear regression, specify regression equation and the dataframe as the function input
model.l = lm(y~x1+x2+x3, data=data.sample)
# take a look at the model result
summary(model.l)
```

- Take a look at the coefficient estimates to see whether your model captured the true relationships.

Step 4: Simple Linear Regression (cont.)

- Take a further look at the regression result:

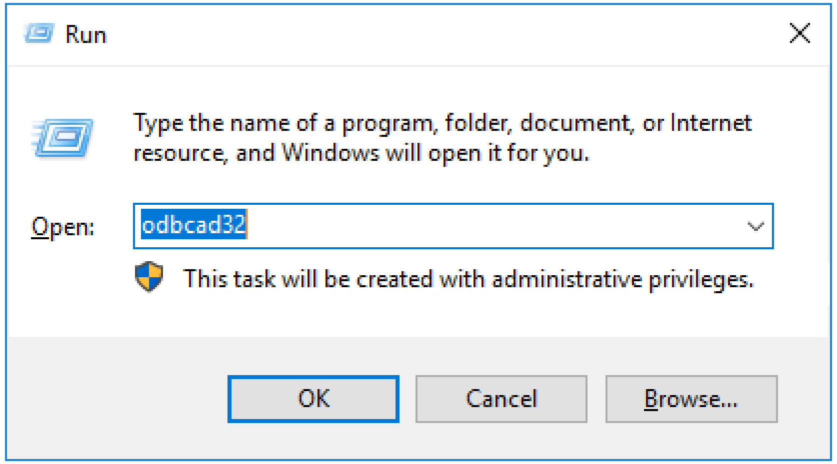
```
# add the model prediction to the dataframe as a new column
# a new column will be automatically created when you assign values into the column
data.sample$y.pred = model.l$fitted.values
# plot the predictions against observations
plot(data.sample$y, data.sample$y.pred)

# this statement will change the Graphics Window layout to show four sub-figures (in a 2x2 grid)
par(mfrow=c(2,2))
# plot some residual graphs
plot(model.l)
# reset the Graphics Window layout
par(mfrow=c(1,1))
```

- The graphs created here can help you test whether your residuals are unbiased/follow a normal distribution. You don't need to worry about the interpretations.

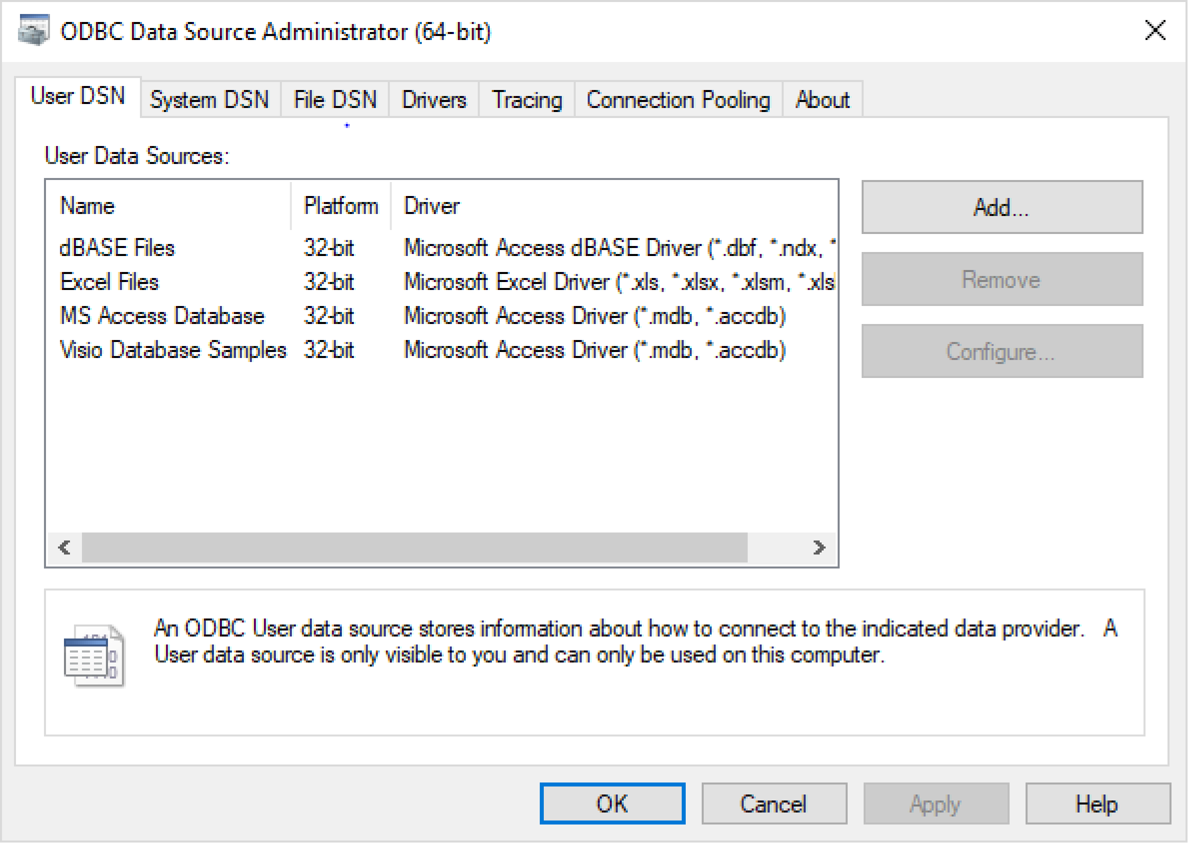
Step 5: Create a Data Source Name (DSN)

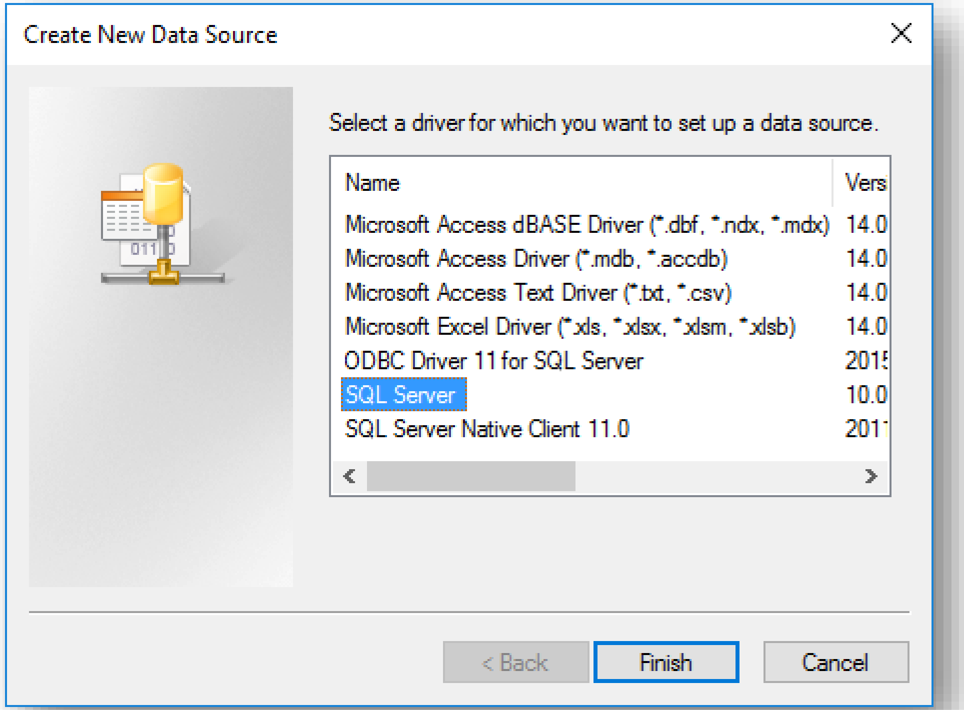
- DSN is a file that defines the connection to a particular database.
- A DSN contains connection information such as the database name, user login and password.
- Click **Start → Run** or Win+R and type "odbcad32"



Step 5: Create a DSN (cont.)

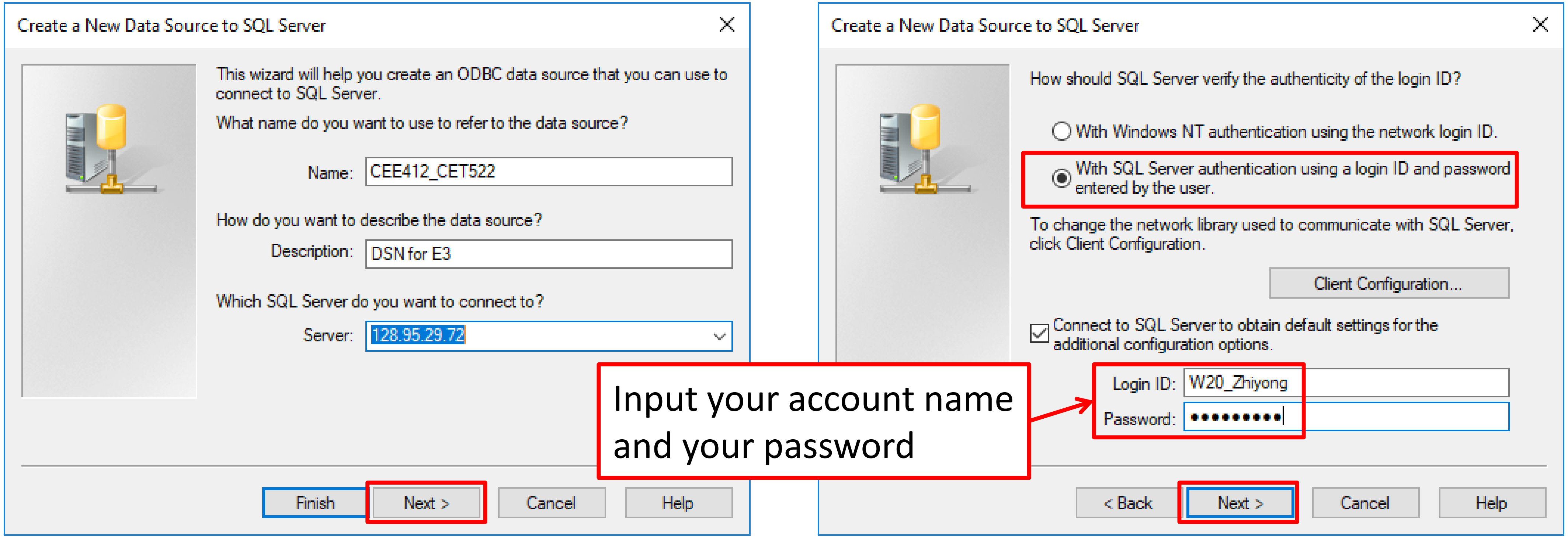
- Click **Add** to add a new DSN, select SQL Server and click **Finish**.





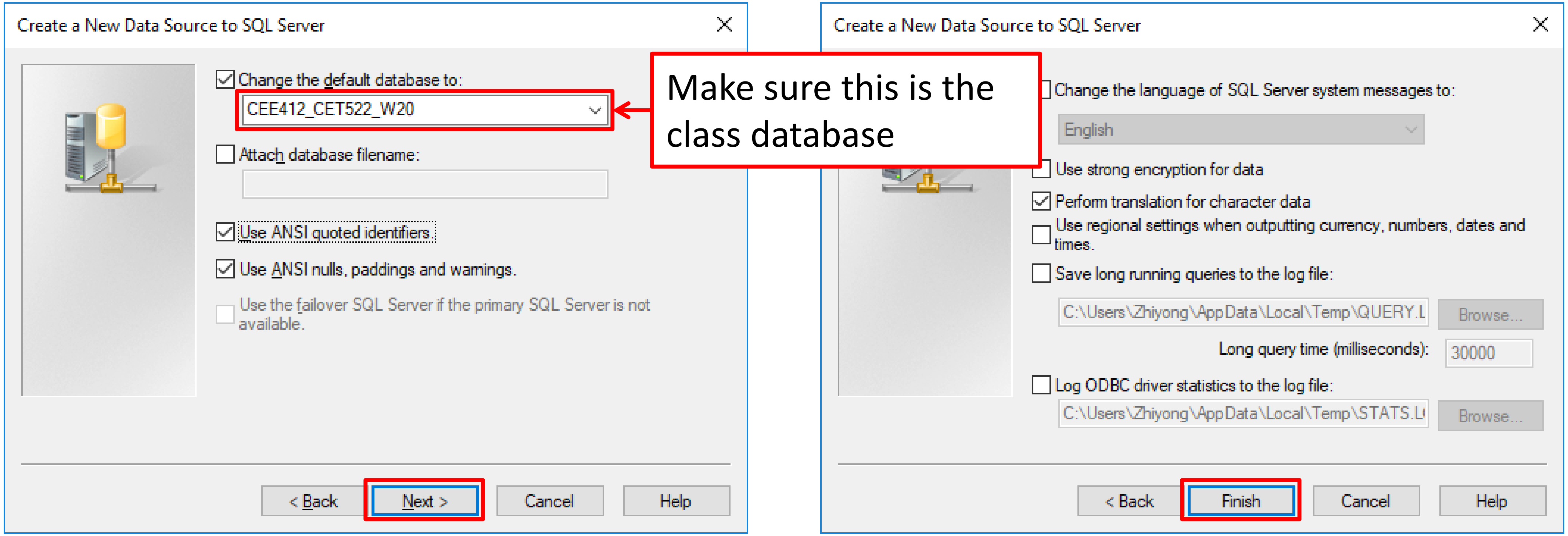
Step 5: Create a DSN (cont.)

- Name your DSN something you will remember, use the class database IP, then click **Next**.
- Select SQL Server authentication and enter your SQL Server Login information, then click **Next**.



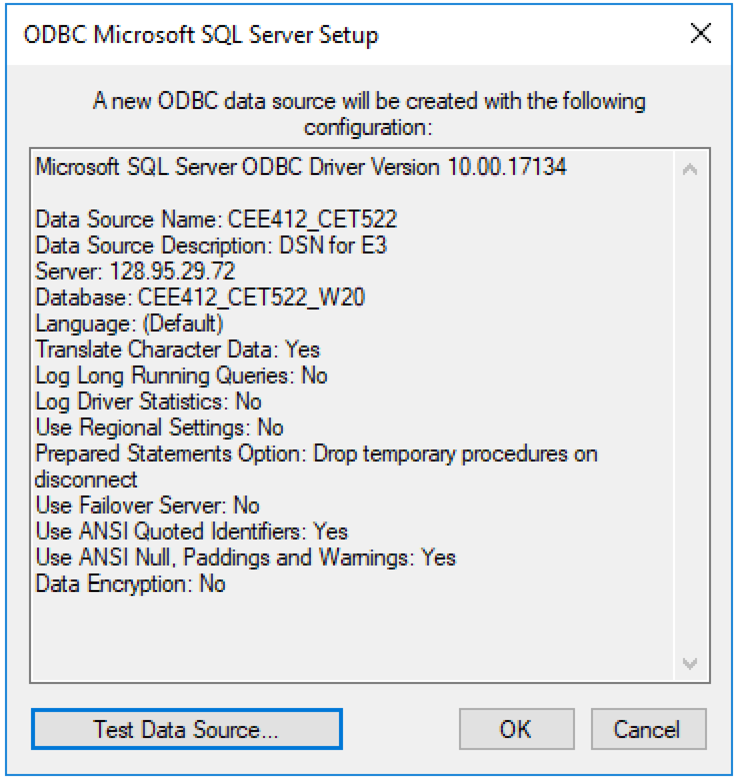
Step 5: Create a DSN (cont.)

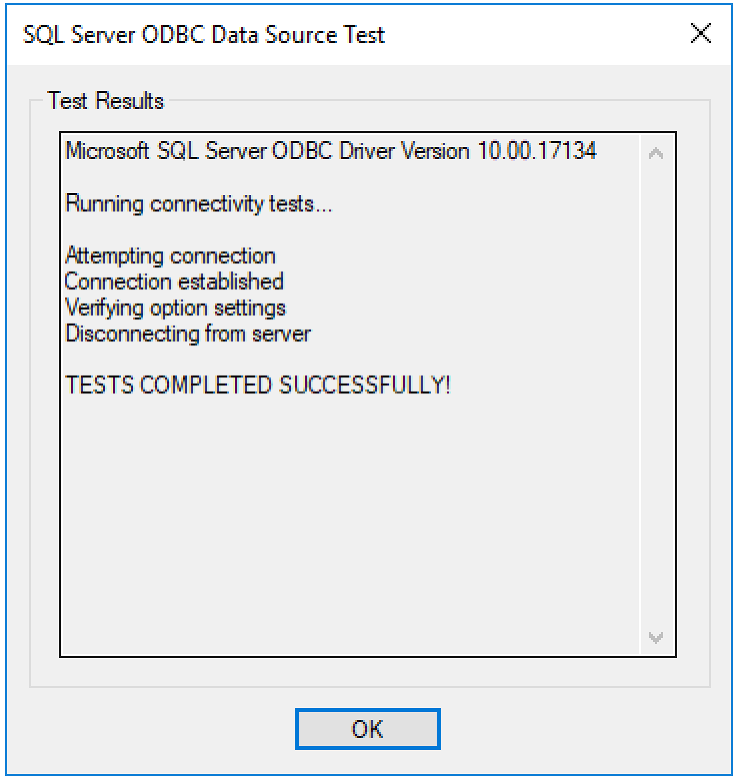
- Make sure the class database is the default database, leave everything else to default. Click **Next** and then **Finish**.



Step 5: Create a DSN (cont.)

- You will see a summary of the connection, click "Test Data Source" to see if everything worked, then click **OK**.



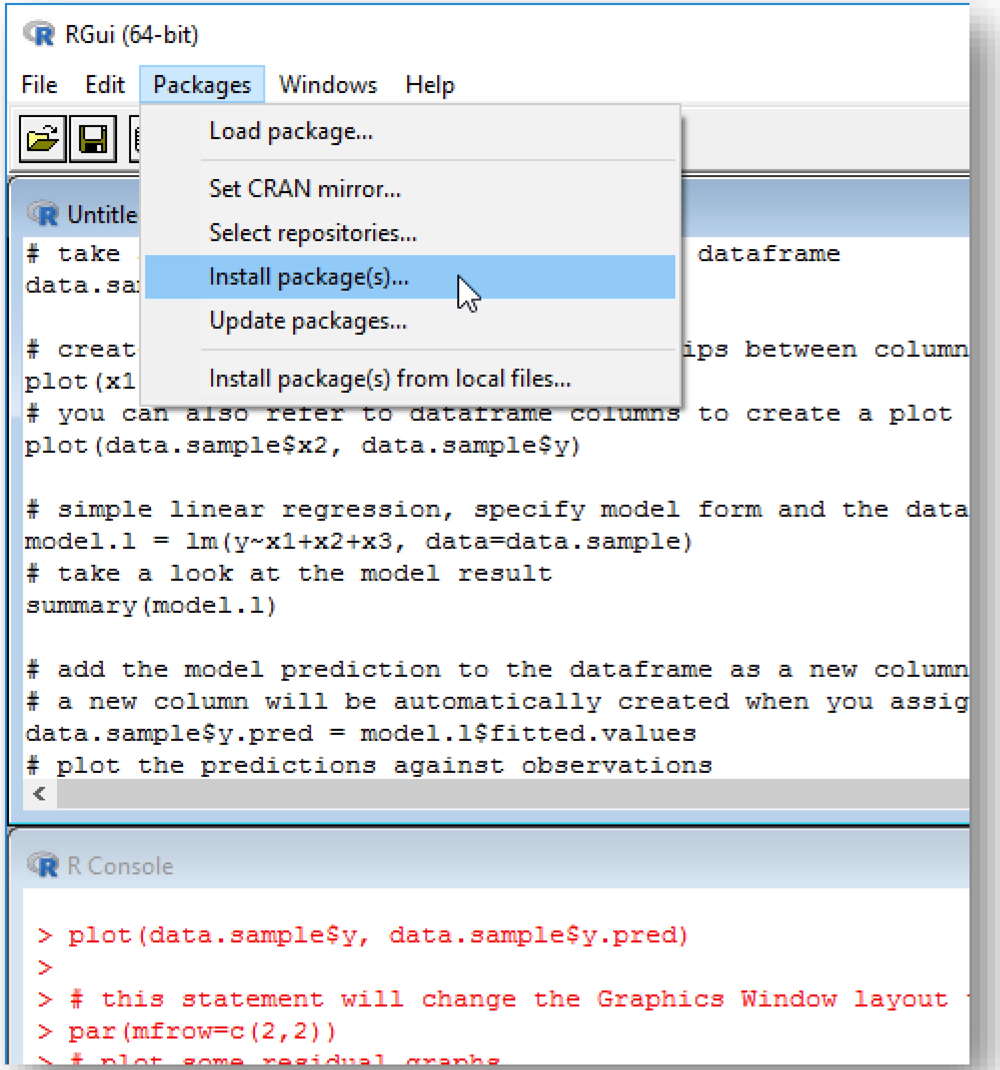


Step 5: Install the RODBC package in R

- There are two ways to do this:
  1. Click **Packages → Install Package(s)** as shown
     - If you are prompted to select a mirror, just choose one close to you
     - Find "RODBC" in the packages list
  2. Use the install.packages() function

```
install.packages("RODBC")
```

  - If this fails, just use the first option.



Step 6: Connect to your DSN

- Load the RODBC package using the library() function:

```
library(RODBC)
```

- Enter the code shown below to create a connection object. You should change the function parameters based on your DSN and SQL Server login

```
conn <- odbcConnect("CEE412_CET522", "Username", "Password")
```

DSN

SQL Server login

Step 6: Connect to your DSN

- Get a list of tables in the database:

```
sqlTables(conn, tableType="TABLE")
```

- You should see something similar to what is shown below:

	TABLE_CAT	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	REMARKS
1	CEE412_CET522_W20	dbo	A4_AccidentCount	TABLE	<NA>
2	CEE412_CET522_W20	dbo	A4_LoopData	TABLE	<NA>
3	CEE412_CET522_W20	dbo	A4_RoadData	TABLE	<NA>
4	CEE412_CET522_W20	dbo	E1_STcabinets	TABLE	<NA>
5	CEE412_CET522_W20	dbo	E1_STloopdat	TABLE	<NA>
6	CEE412_CET522_W20	dbo	E2_CEOs	TABLE	<NA>
7	CEE412_CET522_W20	dbo	E2_Companies	TABLE	<NA>
8	CEE412_CET522_W20	dbo	E2_Countries	TABLE	<NA>
9	CEE412_CET522_W20	dbo	E2_Cyclists	TABLE	<NA>
10	CEE412_CET522_W20	dbo	E2_Locations	TABLE	<NA>
11	CEE412_CET522_W20	dbo	E2_Weather	TABLE	<NA>
12	CEE412_CET522_W20	dbo	E3_Accident	TABLE	<NA>
13	CEE412_CET522_W20	dbo	E3_Road	TABLE	<NA>
14	CEE412_CET522_W20	sys	trace_xe_action_map	TABLE	<NA>
15	CEE412_CET522_W20	sys	trace_xe_event_map	TABLE	<NA>

Step 6: Query SQL from R

- We will be using tables E3_Accident and E3_Road in the class database.
- The function you will use is sqlQuery(), and the function parameters are shown as follows:

Your connection object

```
data = sqlQuery(conn, "SELECT * FROM table_name")
```

The result dataframe in R

Query statement as a string

Step 6: Query SQL from R

- Write two queries selecting all from the two tables of interest and assigning the resulting dataframes to named objects as shown:

```
Accident = sqlQuery(conn, "SELECT * FROM E3_Accident")
# take a look at the first 5 rows
Accident[1:5,]

Road = sqlQuery(conn, "SELECT * FROM E3_Road")
# take a look at the first 5 rows
Road[1:5,]
```

- The SegID columns in both tables define the relationship, which tells the exact road segment where each accident happens.

Step 7: Format Data for Analysis

- We will perform an accident Hotspot Identification (HSID) analysis introduced in Wednesday's lecture.
- In order to model the relationship between number of accidents and roadway/traffic conditions, we need to count the number of accidents happened on each road segment.
- Below is the SQL code that can create a table for HSID analysis:

```
SELECT r.RouteNo, r.BeginMP, r.EndMP, r.AADT,
       CAST(r.EndMP-r.BeginMP AS DECIMAL(10,2)) AS Length,
       r.SpeedLMT, r.TruckRate,
       COUNT(a.CaseNo) AS AccCnt
  FROM E3_Accident AS a RIGHT JOIN E3_Road AS r
       ON a.SegID = r.SegID
 GROUP BY r.RouteNo, r.BeginMP, r.EndMP, r.AADT, r.EndMP-r.BeginMP,
       r.SpeedLMT, r.TruckRate
```

Step 7: Format Data for Analysis

- You can type the long query statement in R and save it as a string object.
- Then in the sqlQuery() function, use the query string as an input parameter.

```
AccQuery = "SELECT r.RouteNo, r.BeginMP, r.EndMP, r.AADT,
                    CAST(r.EndMP-r.BeginMP AS DECIMAL(10,2)) AS Length,
                    r.SpeedLMT, r.TruckRate,
                    COUNT(a.CaseNo) AS AccCnt
               FROM E3_Accident AS a RIGHT JOIN E3_Road AS r
                    ON a.SegID = r.SegID
              GROUP BY r.RouteNo, r.BeginMP, r.EndMP, r.AADT, r.EndMP-r.BeginMP,
                    r.SpeedLMT, r.TruckRate"

AccCnt_Table = sqlQuery(conn, AccQuery)
```

Step 7: Format Data for Analysis

- Take a look at what you have created.

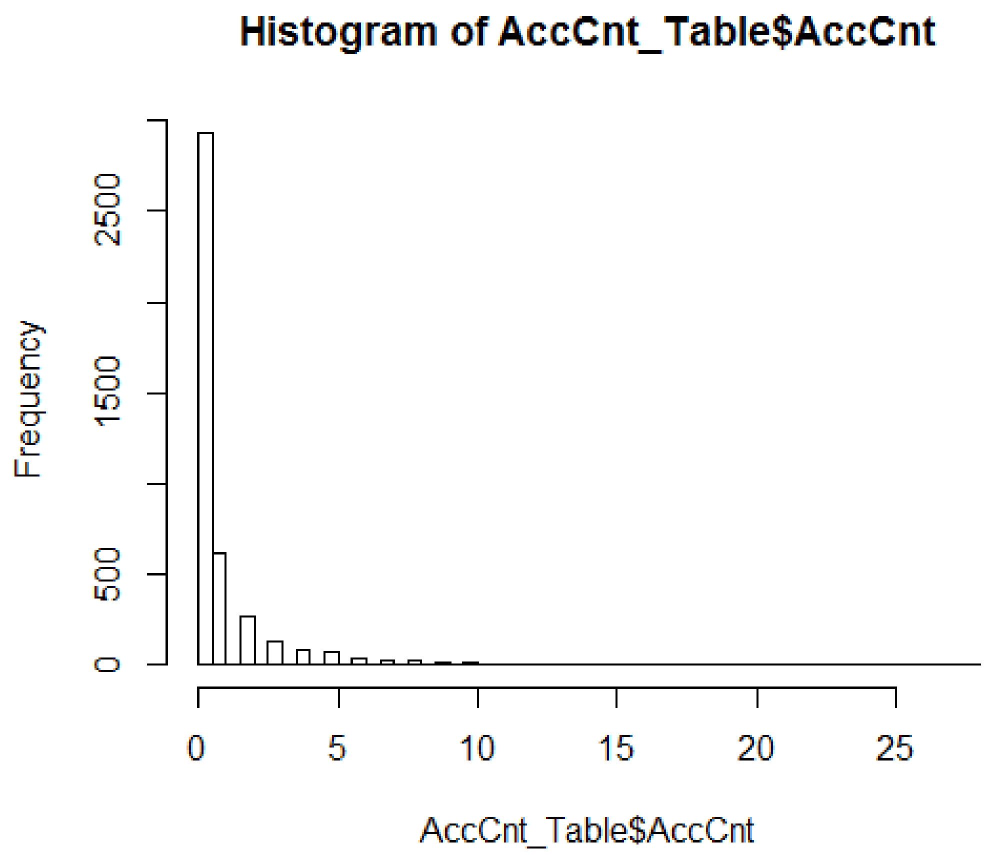
```
# show the first 15 rows of the table, including all columns
AccCnt_Table[1:15,]
```

	RouteNo	BeginMP	EndMP	AADT	Length	SpeedLMT	TruckRate	AccCnt
1	5	0.00	0.27	123000	0.27	50	9	14
2	5	0.27	0.28	123000	0.01	50	0	1
3	5	0.28	0.29	123000	0.01	50	0	3
4	5	0.29	0.32	123000	0.03	50	0	1
5	5	0.32	0.39	123000	0.07	50	0	6
6	5	0.39	0.50	123000	0.11	50	0	3
7	5	0.50	0.59	123000	0.09	50	0	2
8	5	0.59	0.60	123000	0.01	50	0	0
9	5	0.60	0.68	123000	0.08	50	0	1
10	5	0.68	0.69	123000	0.01	50	0	0
11	5	0.69	0.78	123000	0.09	50	8	1
12	5	0.78	0.79	123000	0.01	50	8	0
13	5	0.79	0.82	115958	0.03	50	0	2
14	5	0.82	0.87	115958	0.05	50	0	1
15	5	0.87	1.05	115958	0.18	50	0	2

Step 7: Format Data for Analysis

- See some summary of the accident count data

```
# create a histogram of accident count with 50 bins
hist(AccCnt_Table$AccCnt, breaks = 50)
```



Step 8: Build Regression Models

- It is assumed that several factors may affect the number of accident on roadway, such as truck rate, AADT, speed limit, etc.
- Let us perform two regressions to appropriately model this relationship.
- Model 1: Poisson Regression

```
# fit a Poisson regression model
model.pois = glm(AccCnt~TruckRate+SpeedLMT+AADT, family="poisson", data=AccCnt_Table)
# show a summary of model result
summary(model.pois)
```

  - glm(): function to fit generalized linear models.
  - AccCnt (accident count) is the response variable.
  - Response is separated from predictors by “~”, and all predictors are separated by “+”.
  - No need to use “$” to reference the columns in the dataframe, because the dataframe is specified in “data=AccCnt_Table”.

Step 8: Build Regression Models

- Model 1: Poisson Regression - Result

```
Call:
glm(formula = AccCnt ~ TruckRate + SpeedLMT + AADT, family = "poisson",
    data = AccCnt_Table)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.0563  -0.9761  -0.6442  -0.3810  10.4869

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.421e+00  2.246e-01  10.779  < 2e-16 ***
TruckRate   -8.526e-03  2.518e-03  -3.386 0.000708 ***
SpeedLMT    -6.273e-02  3.663e-03 -17.124  < 2e-16 ***
AADT         1.264e-05  2.616e-07  48.317  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 12906.8  on 4289  degrees of freedom
Residual deviance:  8705.7  on 4286  degrees of freedom
  (2 observations deleted due to missingness)
AIC: 12335
```

Step 8: Build Regression Models

- Model 2: Negative Binomial Regression

```
# load the MASS package
library(MASS)
# fit a negative binomial regression model
model.nb = glm.nb(AccCnt~TruckRate+SpeedLMT+AADT, data=AccCnt_Table)
# show a summary of model result
summary(model.nb)
```

- glm() is a function in the MASS package that can fit negative binomial generalized linear models
- Make sure to load the MASS package before running this model (most likely it’s already installed, so you just need to load the package).

Step 8: Build Regression Models

- Model 2: Negative Binomial Regression - Result

```
Call:
glm.nb(formula = AccCnt ~ TruckRate + SpeedLMT + AADT, data = AccCnt_Tab$
    init.theta = 0.512190089, link = log)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8066  -0.7331  -0.5300  -0.2435   4.3482

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.626e+00  4.175e-01   6.290 3.17e-10 ***
TruckRate   -3.805e-03  3.946e-03  -0.964    0.335
SpeedLMT    -7.127e-02  6.477e-03 -11.003  < 2e-16 ***
AADT         1.493e-05  5.572e-07  26.790  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.5122) family taken to be 1)

    Null deviance: 4616.6  on 4289  degrees of freedom
Residual deviance: 2963.1  on 4286  degrees of freedom
  (2 observations deleted due to missingness)
AIC: 9159.1
```

Step 9: Analyze Model Results

Which model performs better for this accident dataset?

We can use the Akaike Information Criterion (AIC) to compare the two models.

- AIC offers a relative estimate of the information lost for a given model.
- It considers both the goodness-of-fit and the model complexity.
- The model with the minimum AIC value is preferred.

$$AIC = 2k - 2\ln(\hat{L})$$

where,

k = number of parameters in the model

$\hat{L}$ = maximized value of the likelihood function for the estimated model

Step 9: Analyze Model Results

- You can find the AIC values in the model summaries.
- Or, you can use the AIC() function as shown:

```
# find the AIC values for two models
AIC(model.pois)
AIC(model.nb)
```

12335
9159

- Negative Binomial model is preferred for the studied accident data.

- You can reference individual model elements as shown:

```
# get the dispersion parameter in the NB model
model.nb$theta
```