

Eric Pitman Summer Workshop in Computational Science

Intro to 
RStudio Tips

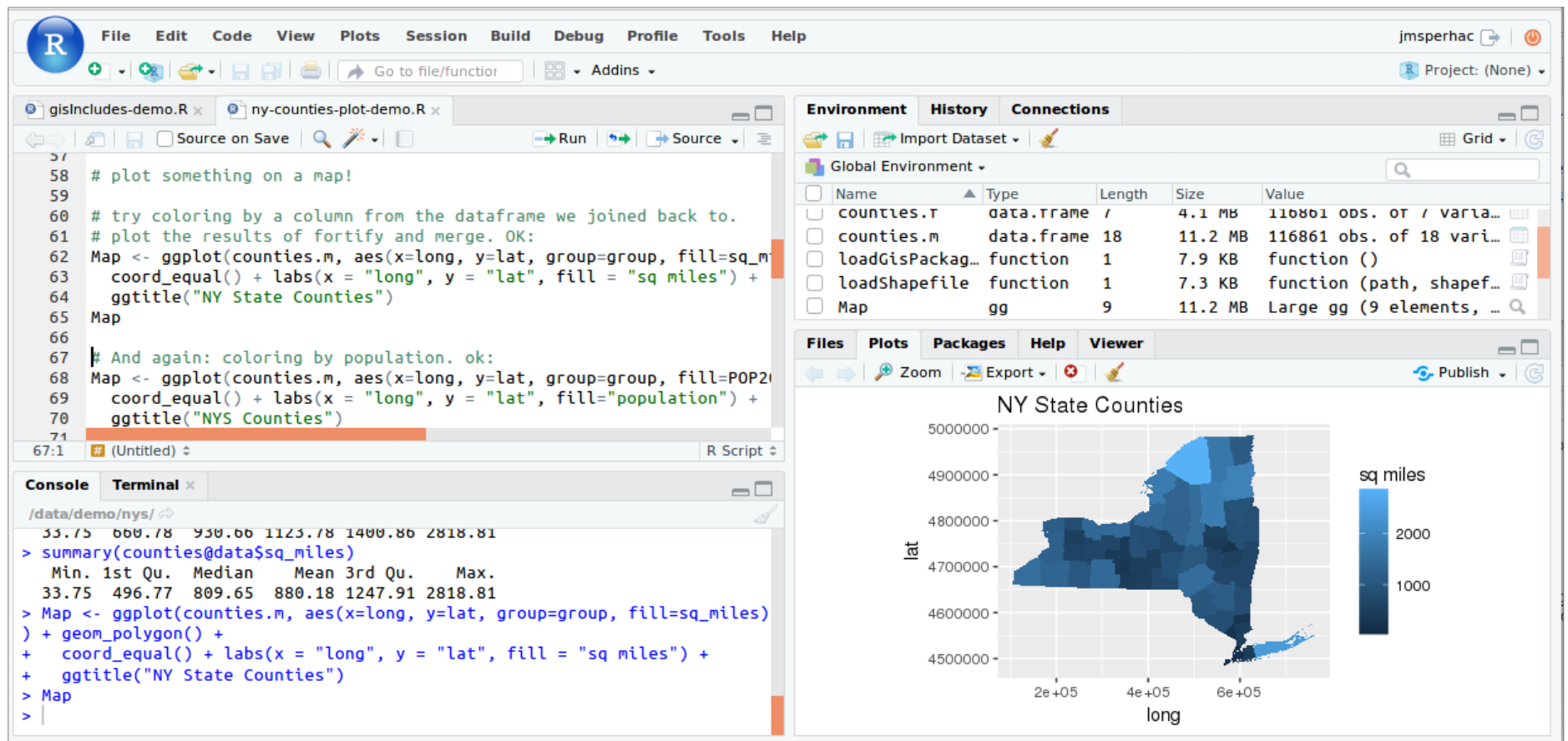


CENTER FOR **COMPUTATIONAL RESEARCH**



University at Buffalo
The State University of New York

RStudio



Four Rstudio panes: Editor, Environment/History, Plots/Help, Console

R Practical Matters

for RStudio



- R is case sensitive (R != r)
- Command line prompt is `>`
- To run R code: use command line, or save script and `source("script_name")`
- To separate commands, use `;` or a newline
- The `#` character marks a non-executed *comment*
- To display help files:
`?<command-name>` or `??<command-name>`



RStudio basics and tips

- Up-arrow and history pane: access and edit previous commands
- You can change window size in the IDE by dragging window borders
- Ctrl-L clears the console window
- Broom icon clears Workspace or Plots
- Is your Project loaded? Check upper right.

VIDIA Dashboard

The screenshot displays the VIDIA Dashboard interface. At the top, a blue header bar contains the user's name 'J M Sperhac' and a 'Dashboard' link. A sidebar on the left lists navigation options: Dashboard (selected), Profile, Groups (17), Account, Contributions (34), Usage, Collections, Messages (132), Projects (2), Citations, and Activity (867). The main content area is divided into four panels:

- My Tools:** A list of tools with heart icons for favorites and folder icons for organization. The tools listed are GNU Octave IDE, IPython QT Console, Jupyter, Orange, PSPP, Rapid Miner v5, RStudio, and Spyder Python IDE. Below the list is a note: 'Add a tool to your favorites by clicking a heart. Click the heart again to remove it.'
- Tips and Documentation:** Contains sections for 'VIDIA Tips' (with links to Knowledge Base, Using VIDIA, and HUBzero user documentation), 'Tools and Tool Documentation' (with links to R and RStudio, RapidMiner, and PSPP), and 'File Access' (with links to Upload and download your files, and File access with UBBBox).
- My Sessions:** Displays a thumbnail image of a session and the text 'LAST ACCESSED: June 06, 2018 @ 3:45pm'. Below this are 'Open' and 'Terminate' buttons.
- Dashboard Introduction:** A text box with a welcome message and instructions on how to use the dashboard, including the 'Add Modules' button and the ability to remove or rearrange modules.

VIDIA Dashboard: RStudio Tool

The screenshot displays the VIDIA Dashboard interface for user J M Sperhac. The dashboard is organized into several sections:

- Left Sidebar:** Contains navigation links for Dashboard, Profile, Groups (17), Account, Contributions (34), Usage, Collections, Messages (132), Projects (2), Citations, and Activity (867).
- Top Header:** Shows the user's name 'J M Sperhac' and a 'Dashboard' tab, along with an 'Add Modules' button.
- My Tools Section:** A list of installed tools including GNU Octave IDE, IPython QT Console, Jupyter, Orange, PSPP, Rapid Miner v5, **RStudio** (highlighted with a red circle and a red arrow), and Spyder Python IDE. Each tool has a heart icon for favoriting and a trash icon for removal.
- Tips and Documentation Section:** Provides links for VIDIA Tips (Knowledge Base, Using VIDIA, HUBzero user documentation) and Tools and Tool Documentation (R and RStudio, RapidMiner).
- My Sessions Section:** Displays a session thumbnail, the last accessed time (June 06, 2018 @ 3:45pm), and buttons to 'Open' or 'Terminate' the session.
- Dashboard Introduction Section:** Offers a welcome message and instructions on how to use the dashboard, including the 'Add Modules' button.

RStudio Interactive Development Environment (IDE)

1. Editor

The screenshot displays the RStudio IDE interface. The top pane is the Editor, showing an R script with a function `makePie` and its execution in the Console. The right pane is split into the Environment pane (showing the 'cr' data frame) and the Plots pane (showing a pie chart titled 'Vehicle Drivetrain Type in Cars93 Dataset').

Environment Pane:

Variable	Value
celsius	int [1:6] 20 21 22 23 24 25
lbls	chr [1:6] "3 Cylinders: 3%" "4 Cylinders: 3%"
pct	table [1:6(1d)] 3 53 2 33 8 1
slices	'table' int [1:6(1d)] 3 49 2 31 7 1

Plots Pane:

Vehicle Drivetrain Type in Cars93 Dataset

Drivetrain Type	Percentage
Front	72%
4WD	11%
Rear	17%

Console:

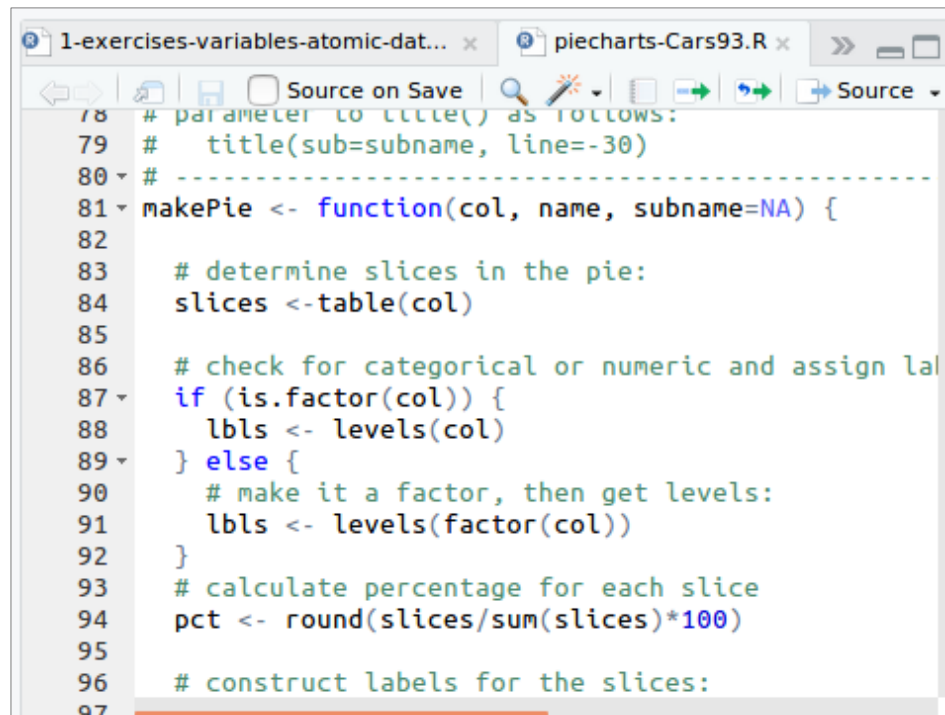
```
> # ----- a few examples using the makePie() function -----  
> #  
> makePie(cr$Type, "Vehicle Type")  
  
> makePie(cr$DriveTrain, "Vehicle Drivetrain Type")  
  
> makePie(cr$Turn.circle, "Turn Circle", "U-Turn space (feet)")  
  
> #makePie(cr$Passenger, "Max Number of Passengers")  
> #makePie(cr$Cylinders, "# Cylinders")  
> #makePie(cr$AirBags, "Vehicle Airbags")  
> #makePie(cr$O .... [TRUNCATED]  
>
```

2. Workspace (Variables) and History

3. Plots, etc.

4. Console

Editor window

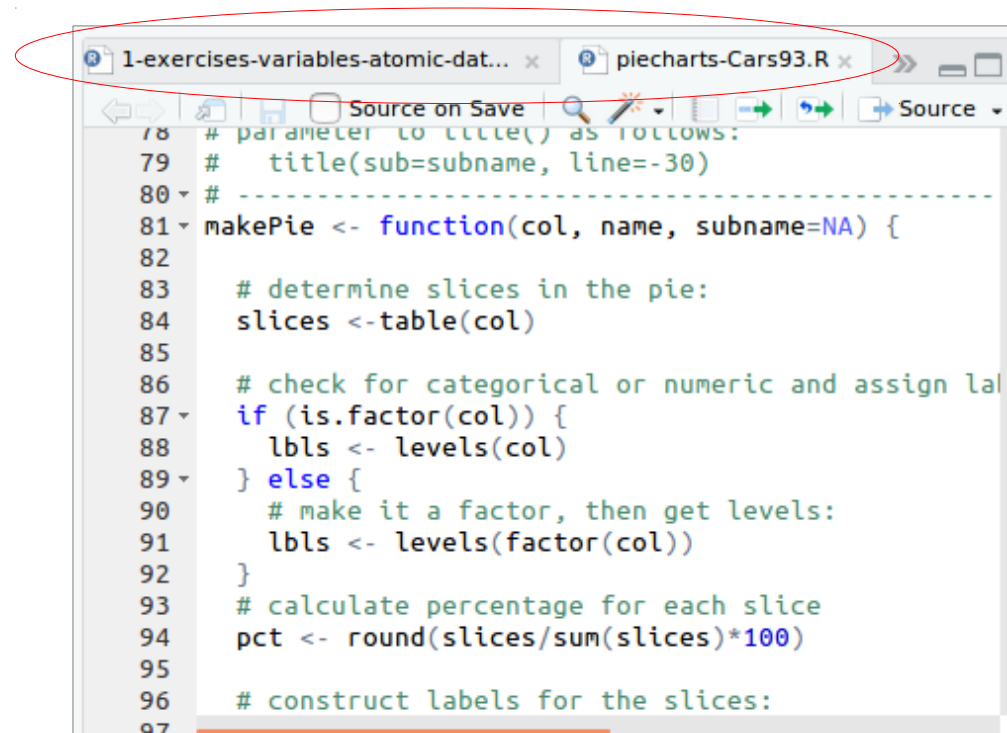


The image shows a screenshot of an R Studio editor window. The window has two tabs at the top: '1-exercises-variables-atomic-dat...' and 'piecharts-Cars93.R'. The 'piecharts-Cars93.R' tab is active. Below the tabs is a toolbar with various icons for file operations, search, and execution. The main area of the window contains R code. The code defines a function 'makePie' that takes three arguments: 'col', 'name', and 'subname'. The function determines the slices of the pie, checks for categorical or numeric data, calculates the percentage for each slice, and constructs labels for the slices. The code is as follows:

```
78 # parameter to title() as follows:
79 #   title(sub=subname, line=-30)
80 # -----
81 makePie <- function(col, name, subname=NA) {
82
83   # determine slices in the pie:
84   slices <- table(col)
85
86   # check for categorical or numeric and assign labels
87   if (is.factor(col)) {
88     lbls <- levels(col)
89   } else {
90     # make it a factor, then get levels:
91     lbls <- levels(factor(col))
92   }
93   # calculate percentage for each slice
94   pct <- round(slices/sum(slices)*100)
95
96   # construct labels for the slices:
97
```

Select, view, edit, save, and execute scripts.

Editor window



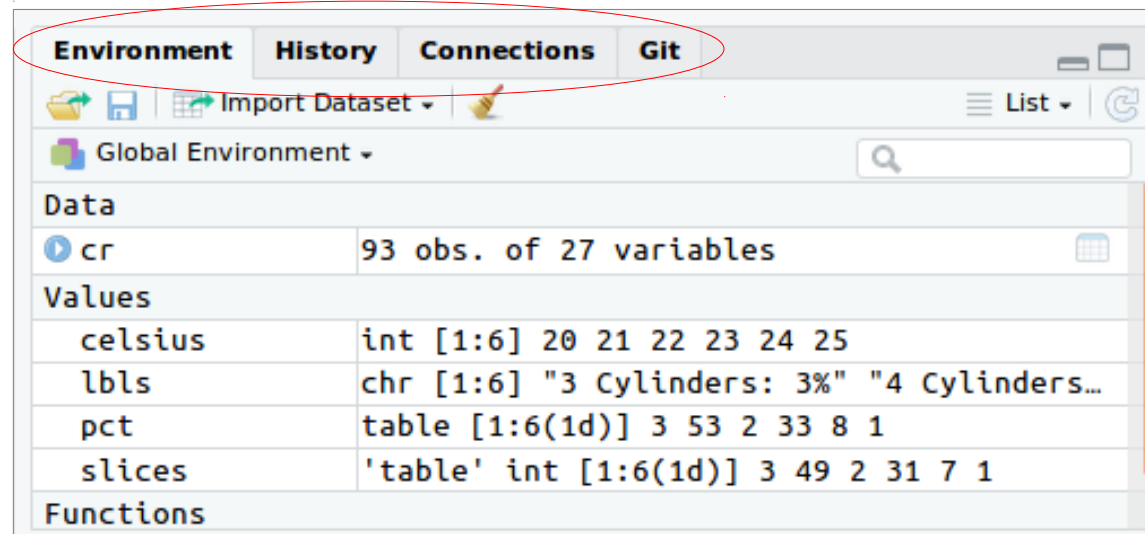
```
78 # parameter to title() as follows:
79 #   title(sub=subname, line=-30)
80 # -----
81 makePie <- function(col, name, subname=NA) {
82   # determine slices in the pie:
83   slices <- table(col)
84   # check for categorical or numeric and assign labels
85   if (is.factor(col)) {
86     lbls <- levels(col)
87   } else {
88     # make it a factor, then get levels:
89     lbls <- levels(factor(col))
90   }
91   # calculate percentage for each slice
92   pct <- round(slices/sum(slices)*100)
93   # construct labels for the slices:
94 }
```

Select, view, edit, save, and execute scripts.

Workspace and History window

Environment		History	Connections	Git
Global Environment		List		
Data				
cr	93 obs. of 27 variables			
Values				
celsius	int [1:6] 20 21 22 23 24 25			
lbls	chr [1:6] "3 Cylinders: 3%" "4 Cylinders..."			
pct	table [1:6(1d)] 3 53 2 33 8 1			
slices	'table' int [1:6(1d)] 3 49 2 31 7 1			
Functions				

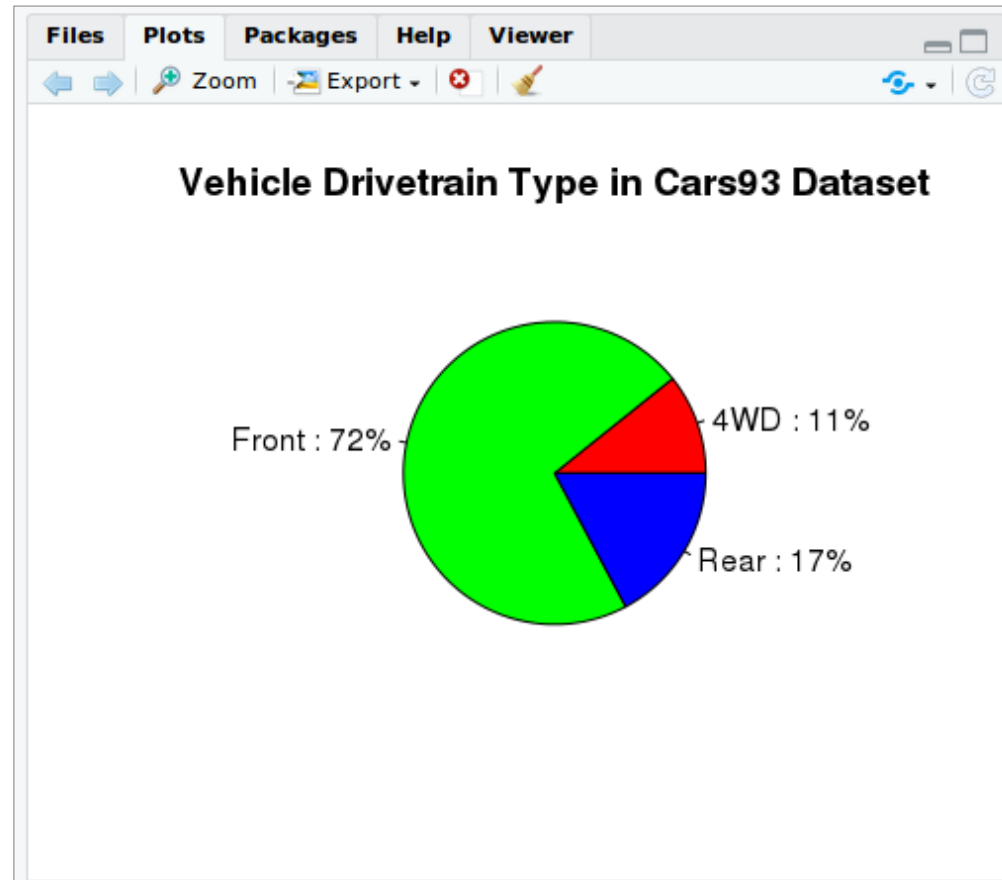
Workspace and History window



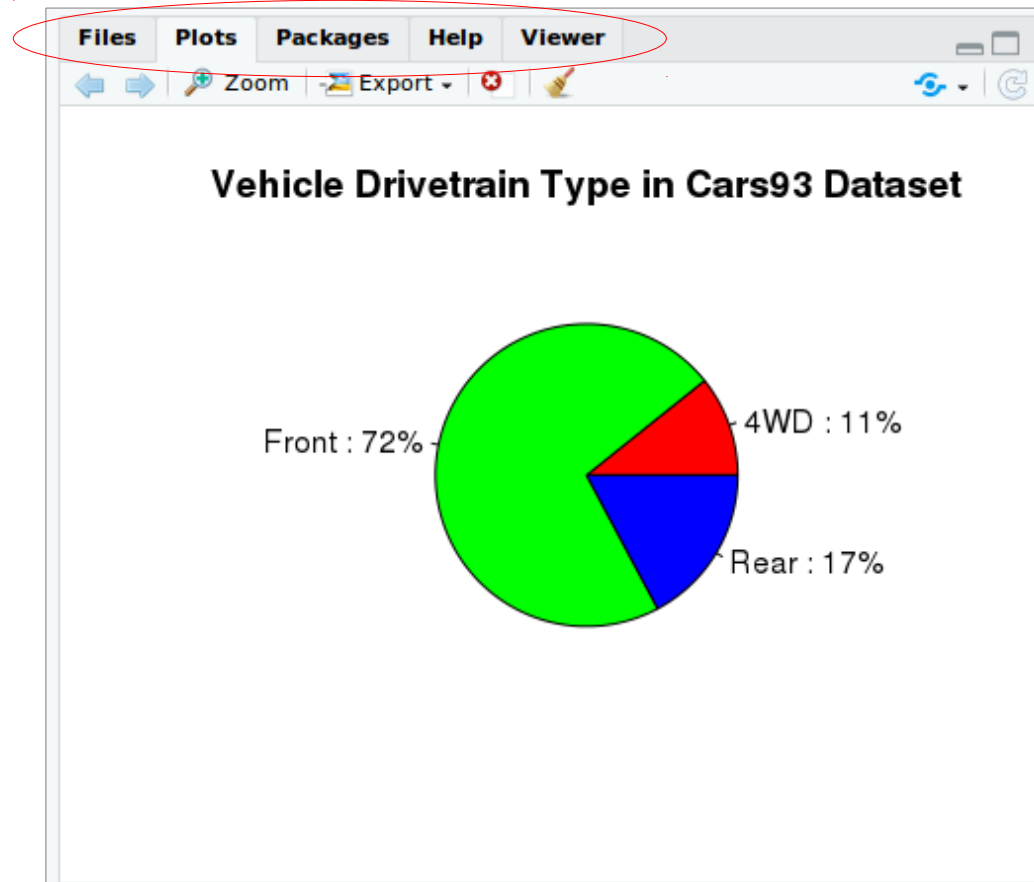
Pick a **tab** to access:

- Current variables (**Environment**)
- Command **History**
- (Database) **Connections**
- Version control (**Git**) status and commands

Plot (etc.) window



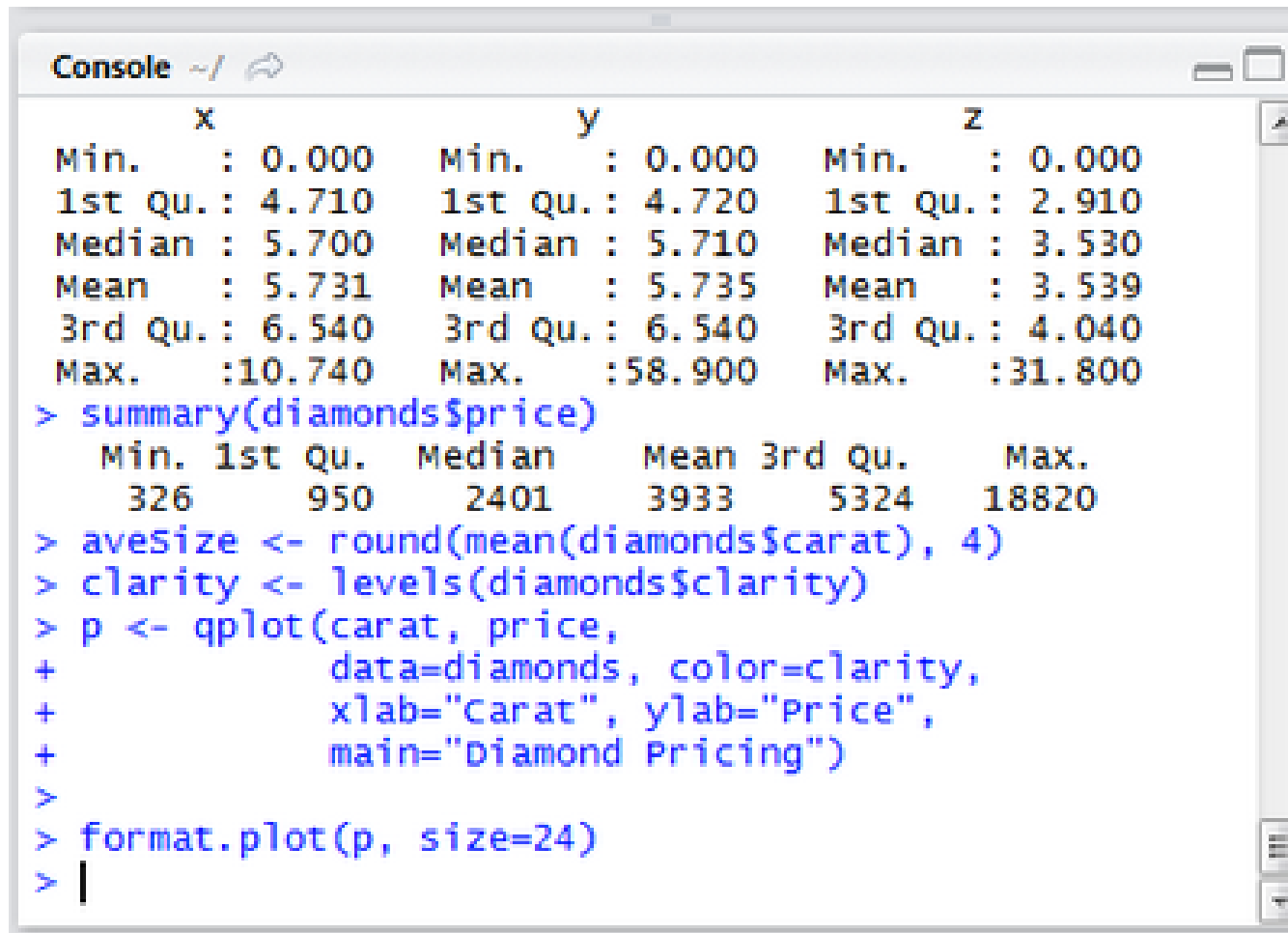
Plot (etc.) window







Pick a **tab** to:

- Access **Files** and Directories
- View current **Plots**
- Review loaded and available **Packages**
- Read **Help** and Documentation
- **View** markdown

Console window

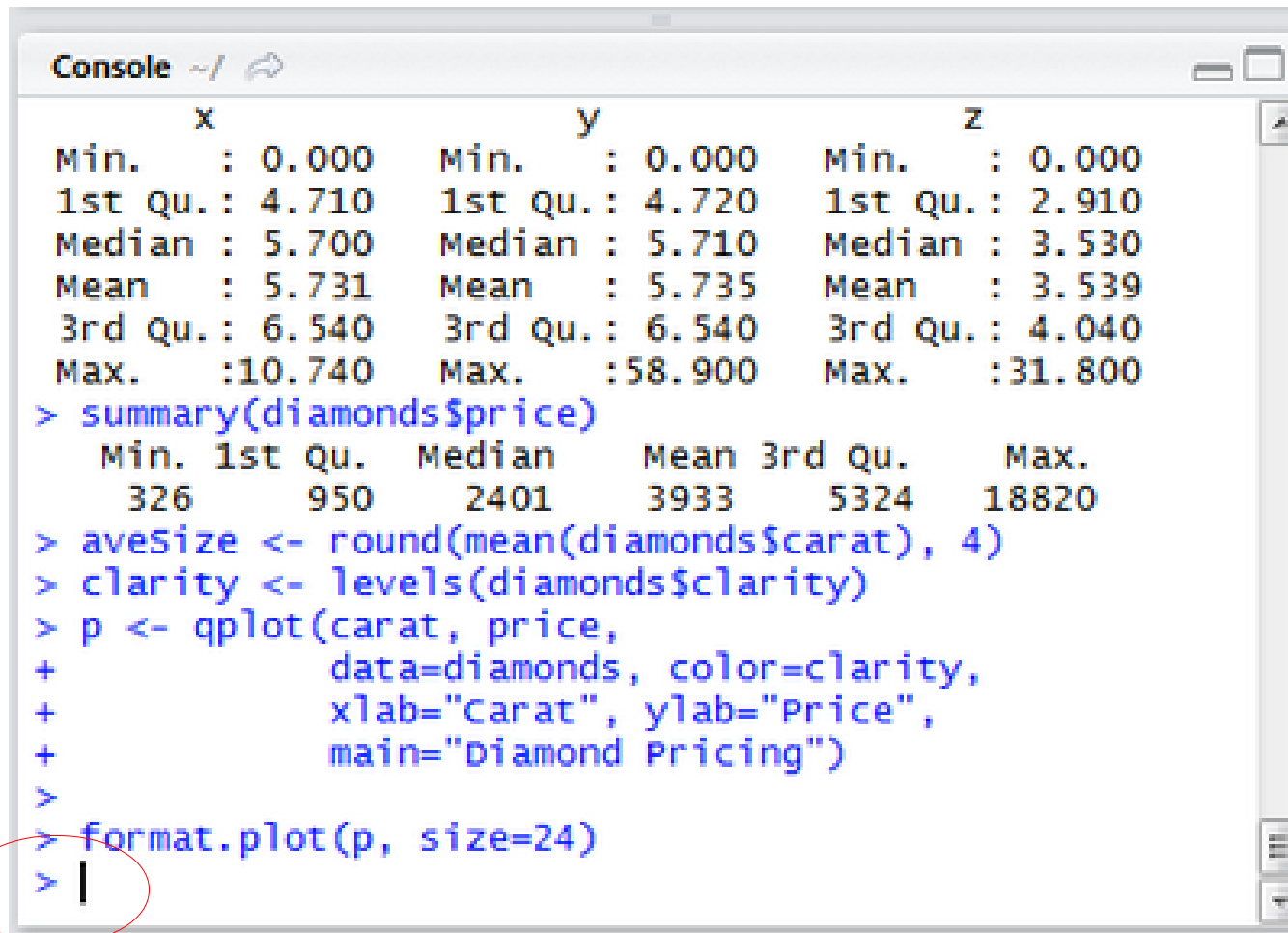



```
Console ~/      
  
      x              y              z  
Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910  
Median : 5.700   Median : 5.710   Median : 3.530  
Mean   : 5.731   Mean   : 5.735   Mean   : 3.539  
3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040  
Max.   :10.740   Max.   :58.900   Max.   :31.800  
> summary(diamonds$price)  
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.        
  326   950   2401   3933   5324   18820  
> aveSize <- round(mean(diamonds$carat), 4)  
> clarity <- levels(diamonds$clarity)  
> p <- qplot(carat, price,  
+           data=diamonds, color=clarity,  
+           xlab="carat", ylab="Price",  
+           main="Diamond Pricing")  
>  
> format.plot(p, size=24)  
> |
```

On the command line:

- Issue commands
- Review the results
- Puzzle over error messages

Console window



```
Console ~/ 
      x              y              z
Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
Median : 5.700   Median : 5.710   Median : 3.530
Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  326    950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+           data=diamonds, color=clarity,
+           xlab="Carat", ylab="Price",
+           main="Diamond Pricing")
>
> format.plot(p, size=24)
> |
```

Command
prompt

On the command line:

- Issue commands
- Review the results
- Puzzle over error messages

Quitting RStudio

Screenshot of RStudio interface showing the process of quitting the application.

The top toolbar contains several icons. A red arrow points to the "Quit" icon (a power button symbol) located in the top right corner of the toolbar.

The main editor window displays R code for creating a pie chart:

```
78 # parameter to title() as follows:
79 # title(sub=subname, line=-30)
80 #
81 makePie <- function(col, name, subname=NA) {
82   # determine slices in the pie:
83   slices <- table(col)
84   # check for categorical or numeric and assign labels
85   if (is.factor(col)) {
86     lbls <- levels(col)
87   } else {
88     # make it a factor, then get levels:
89     lbls <- levels(factor(col))
90   }
91   # calculate percentage for each slice
92   pct <- round(slices/sum(slices)*100)
93   # construct labels for the slices:
94 }
```

The Environment pane on the right shows the Global Environment with the following data:

Data	Values
cr	93 obs. of 27 variables
celsius	int [1:6] 20 21 22 23 24 25
lbls	chr [1:6] "3 Cylinders: 3%" "4 Cylinders..."
pct	table [1:6(1d)] 3 53 2 33 8 1
slices	'table' int [1:6(1d)] 3 49 2 31 7 1

The Plots pane on the right displays a pie chart titled "Vehicle Drivetrain Type in Cars93 Dataset". The chart shows the distribution of drivetrain types:

- Front : 72%
- 4WD : 11%
- Rear : 17%

The Console pane at the bottom shows the execution of the `makePie()` function:

```
> # ----- a few examples using the makePie() function call: -----
> #
> makePie(cr$Type, "Vehicle Type")
> makePie(cr$DriveTrain, "Vehicle Drivetrain Type")
> makePie(cr$Turn.circle, "Turn Circle", "U-Turn space (feet)")
> #makePie(cr$Passenger, "Max Number of Passengers")
> #makePie(cr$Cylinders, "# Cylinders")
> #makePie(cr$AirBags, "Vehicle Airbags")
> #makePie(cr$O .... [TRUNCATED]
>
```


Quitting RStudio

The image shows the RStudio interface with a script editor, environment pane, and a pie chart. A dialog box titled "R Session Ended" is overlaid in the center, featuring a person icon and a "Start New Session" button.

Script Editor:

```
78 # parameter to title() as follows:
79 #   title(sub=subname, line=-30)
80 # -----
81 makePie <- function(col, name, subname=NA) {
82
83   # determine slices in the pie:
84   slices <- table(col)
85
86   # check for categorical or numeric and assign labels
87   if (is.factor(col)) {
88     lbls <- levels(col)
89   } else {
90     # make it a factor,
91     lbls <- levels(fact
92   }
93   # calculate percentag
94   pct <- round(slices/s
95
96   # construct labels fo
97   lbls <- paste(lbls, "
98   lbls <- paste(lbls, "
99
100  # plot it:
101  p <- pie(slices,
102
```

Environment Pane:

Global Environment	
cr	93 obs. of 27 variables
lbls	chr [1:6] "3 Cylinders: 3%" "4 Cylinders: ...
pct	table [1:6(1d)] 3 53 2 33 8 1
slices	'table' int [1:6(1d)] 3 49 2 31 7 1

Pie Chart:

pie in Cars93 Dataset

- 4WD : 11%
- Rear : 17%

Console:

```
> # Here is a general makePie(
umber
> # of .... [TRUNCATED]

> # ----- a few examples using the makePie() function c
all: -----
> #
> makePie(cr$Type, "Vehicle Type")

> makePie(cr$DriveTrain, "Vehicle Drivetrain Type")

> makePie(cr$Turn.circle, "Turn Circle", "U-Turn space (feet
)")

> #makePie(cr$Passenger, "Max Number of Passengers")
> #makePie(cr$Cylinders, "# Cylinders")
> #makePie(cr$AirBags, "Vehicle Airbags")
> #makePie(cr$O .... [TRUNCATED]
>
```

Jupyter R notebook

VIDIA jupyter Candy-distributions Last Checkpoint: 03/25/2019 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Reset equation numbering Dashboard View: </>

8	4	3	1	3	1	5
9	5	2	0	4	3	5
10	2	3	3	6	4	6

In [97]: *# print the structure of the mm data object. We can see the 1000 rows (observations), with 6 colors for each one.*
`str(mm)`

```
int [1:1000, 1:6] 3 2 2 0 3 3 0 4 5 2 ...  
- attr(*, "dimnames")=List of 2  
..$ : chr [1:1000] "1" "2" "3" "4" ...  
..$ : chr [1:6] "red" "blue" "green" "yellow" ...
```

In [98]: *# print the first 10 row names of the mm data object. These just label the observations:*
`rownames(mm)[1:10]`

```
'1' '2' '3' '4' '5' '6' '7' '8' '9' '10'
```

In [99]: *# print the column names of the mm data object. These are the color names:*
`colnames(mm)`

```
'red' 'blue' 'green' 'yellow' 'orange' 'brown'
```

Command Line Prompt



- RStudio:

>

- Jupyter:

In [*integer*]



...is free

If you want to experiment further with R and RStudio, you can install them on your favorite operating system at home.

First, install R:

<http://cran.r-project.org/>

Then, install the RStudio IDE:

<http://www.rstudio.com/ide/>