

Intermediate RStudio Training

December 13, 2017



Course Outline

- **Productivity features**
- **Essential data skills**
- **Factors**
- **Descriptive statistics**
- **Covariances and correlations**

Productivity in RStudio



RStudio Cheat Sheet

File Edit Code View Plots Session Build Debug Profile Tools **Help**

Go to file/function Addins

Write Code

Navigate tabs

Open in new window

Save

Find and replace

Compile as notebook

Run selected code

Import data with wizard

History of past commands to run/copy

Display .Rpres slideshows **File > New File > R Presentation**

Load workspace

Save workspace

Delete all saved objects

Search inside environment

Choose environment to display from list of parent environments

Display objects as list or grid

Displays saved objects by type with short description

View in data viewer

View function source code

Create folder

Upload file

Delete file

Rename file

Path to displayed directory

A File browser keyed to your working directory. Click on file or directory name to open.

Multiple cursors/column selection with **Alt + mouse drag**.

Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.

Syntax highlighting based on your file's extension

Tab completion to finish function names, file paths, arguments, and more.

Multi-language code snippets to quickly use common blocks of code.

Jump to function in file

Change file type

Working Directory

Press **↑** to see command history

Maximize, minimize panes

Drag pane boundaries

```

1 # Good Start...
2
3
4
5
6 "P0030001"
7 "P0030002"
8 "P0030003"
9 "P0030004"
10
11
12 get_digit <- function() {
13   ("num" %% (10 ^ n))
14   %% (10 ^ (n - 1))
15 }
16
17 fo
18   for {GlobalEnv}
19   foo {base}
20   force {base}
21
22
23 (Top Level)
  
```

Console

```

~/IDEcheatsheet/
> foo(1)
[1] 2
> foo <- function(x) x + 1
> foo(2)
> foo(2)
> foo(1)
  
```

R Support

Environment

History

Build

Git

Presentation

Global Environment

Load workspace

Save workspace

Delete all saved objects

Search inside environment

Choose environment to display from list of parent environments

Display objects as list or grid

Displays saved objects by type with short description

View in data viewer

View function source code

Create folder

Upload file

Delete file

Rename file

Path to displayed directory

A File browser keyed to your working directory. Click on file or directory name to open.

Multi-language code snippets to quickly use common blocks of code.

Jump to function in file

Change file type

Working Directory

Press **↑** to see command history

Maximize, minimize panes

Drag pane boundaries

Console

```

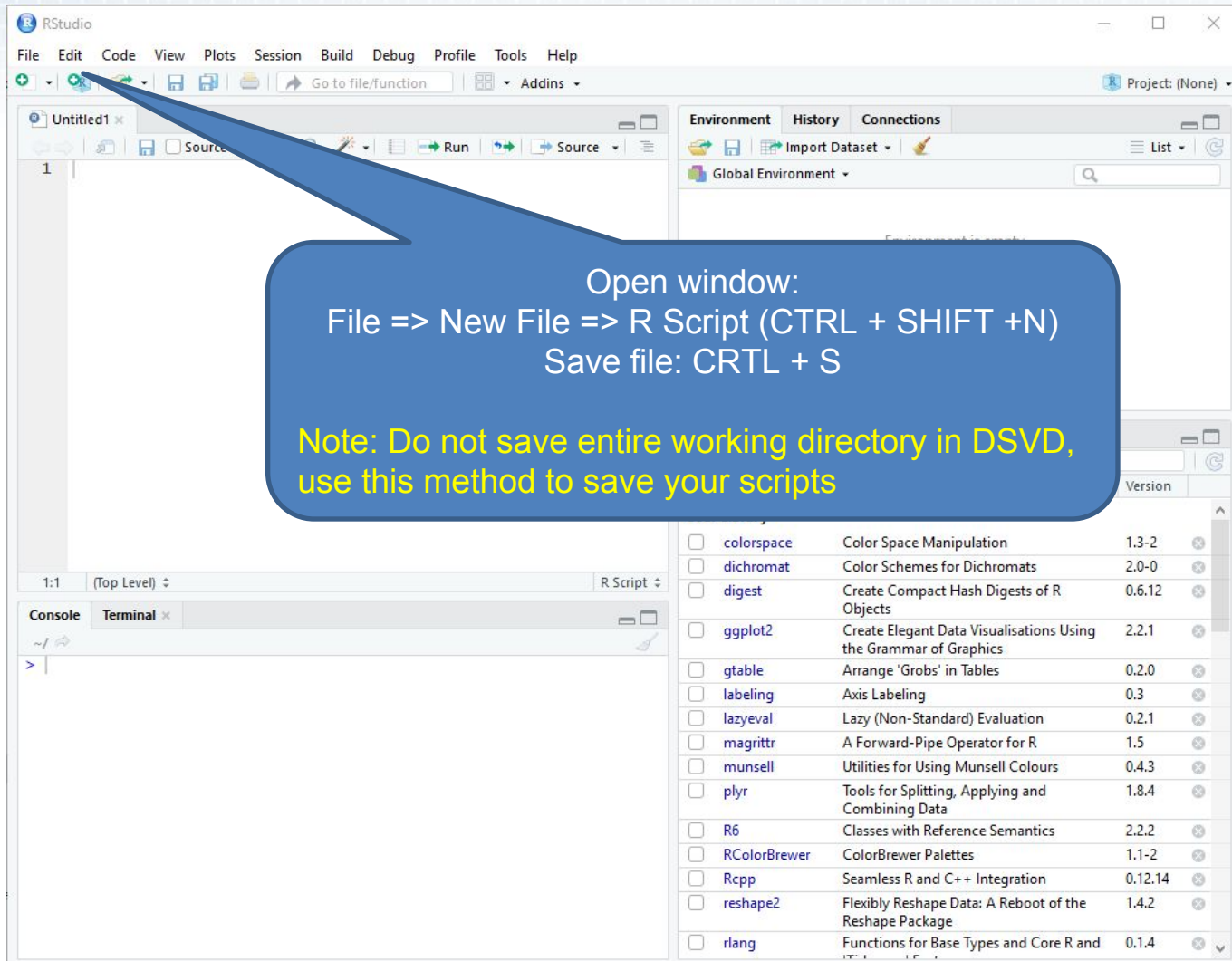
~/IDEcheatsheet/
> foo(1)
[1] 2
> foo <- function(x) x + 1
> foo(2)
> foo(2)
> foo(1)
  
```


Customizing RStudio

The image is a collage of RStudio interface elements, highlighting customization options. The elements are arranged around a central blue arrow pointing towards the bottom right.

- Main Menu Bar:** File Edit Code View Plots Session Build Debug Profile Tools Help
- File Menu:** New File, New Project..., Open File... (Ctrl+O), Recent Files, Open Project..., Open Project in New Session..., Recent Projects, Import Dataset, Save (Ctrl+S), Save As..., Save with Encoding..., Save All (Ctrl+Alt+S), Knit Document (Ctrl+Shift+K), Compile Report..., Print..., Close (Ctrl+W), Close All (Ctrl+Shift+W), Close All Except Current (Ctrl+Alt+Shift+W), Close Project, Quit Session... (Ctrl+Q)
- Help Menu:** R Help, About RStudio, Check for Updates, RStudio Docs, RStudio Support, Cheatsheets (RStudio IDE Cheat Sheet, Data Manipulation with dplyr, tidy, Data Visualization with ggplot2, R Markdown Cheat Sheet, R Markdown Reference Guide, Shiny Web Applications, Package Development with devtools), Keyboard Shortcuts Help (Alt+Shift+K), Markdown Quick Reference, Roxygen Quick Reference, Diagnostics
- Addins Menu:** Show All Panes (Ctrl+Shift+Alt+0), Console on Left, Console on Right, Pane Layout..., Zoom Source (Ctrl+Shift+1), Zoom Console (Ctrl+Shift+2), Zoom Help (Ctrl+Shift+3), Zoom History (Ctrl+Shift+4), Zoom Files (Ctrl+Shift+5), Zoom Plots (Ctrl+Shift+6), Zoom Packages (Ctrl+Shift+7), Zoom Environment (Ctrl+Shift+8), Zoom Viewer (Ctrl+Shift+9), Zoom Connections (Ctrl+Shift+F5)
- Options Dialog Box:** Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.
 - General:** Source, Environment, History, Connect
 - Code:** Environment, History, Files, Plots, Connections, Packages, Help, Build, VCS, Viewer
 - Appearance:** Console, Files, Plots, Packages, Help, View
 - Pane Layout:** Environment, History, Files, Plots, Connections, Packages, Help, Build, VCS, Viewer
- Zoom Shortcuts:** Zoom Source (Ctrl+Shift+1), Zoom Console (Ctrl+Shift+2), Zoom Help (Ctrl+Shift+3), Zoom History (Ctrl+Shift+4), Zoom Files (Ctrl+Shift+5), Zoom Plots (Ctrl+Shift+6), Zoom Packages (Ctrl+Shift+7), Zoom Environment (Ctrl+Shift+8), Zoom Viewer (Ctrl+Shift+9), Zoom Connections (Ctrl+Shift+F5)

Scripting Window



The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar below the menu bar contains icons for creating a new file, opening a file, saving, and running code. A blue callout box with a pointer to the 'New File' icon contains the following text:

Open window:
File => New File => R Script (CTRL + SHIFT + N)
Save file: CTRL + S

Note: Do not save entire working directory in DSVD,
use this method to save your scripts

The background shows the RStudio interface with a script editor on the left, a console at the bottom left, and a package list on the right. The package list includes:

Package	Description	Version
colorspace	Color Space Manipulation	1.3-2
dichromat	Color Schemes for Dichromats	2.0-0
digest	Create Compact Hash Digests of R Objects	0.6.12
ggplot2	Create Elegant Data Visualisations Using the Grammar of Graphics	2.2.1
gttable	Arrange 'Grobs' in Tables	0.2.0
labeling	Axis Labeling	0.3
lazyeval	Lazy (Non-Standard) Evaluation	0.2.1
magrittr	A Forward-Pipe Operator for R	1.5
munsell	Utilities for Using Munsell Colours	0.4.3
plyr	Tools for Splitting, Applying and Combining Data	1.8.4
R6	Classes with Reference Semantics	2.2.2
RColorBrewer	ColorBrewer Palettes	1.1-2
Rcpp	Seamless R and C++ Integration	0.12.14
reshape2	Flexibly Reshape Data: A Reboot of the Reshape Package	1.4.2
rlang	Functions for Base Types and Core R and	0.1.4

Auto-completion in RStudio

- **Tab** is a generic auto-complete function. If you start typing in the console or editor and hit the tab key, RStudio will suggest functions or file names; simply select the one you want and hit either tab or enter to accept it.
- **Control + the up arrow** (command + up arrow on a Mac) is a similar auto-complete tool. Start typing and hit that key combination, and it shows you a list of every command *you've* typed starting with those keys. Select the one you want and hit return. This works only in the interactive console, not in the code editor window.
- **Control + enter** (command + enter on a Mac) takes the current line of code in the editor, sends it to the console and executes it. If you select multiple lines of code in the editor and then hit ctrl/cmd + enter, all of them will run.

Exercise 1: Auto-completion

The screenshot displays the RStudio interface with the following components:

- Environment Panel:** Shows the 'Global Environment' with a variable 'sepal.Length' of type 'num [1:150]' containing values: 5.1, 4.9, 4.7, 4.6, 5, 5.4, 4.6, 5, 4.4, 4.9, ...
- Console:** Contains the following R code:

```
> view(iris)
> hist(iris$sepal.Length)
> sepal.Length = (iris$sepal.Length)
> writ
```

Below the code, an auto-completion menu is open, listing functions: write, write.csv, write.csv2, write.dcf, write.ftable, write.socket, write.table, and writeBin. The 'write.csv' function is selected, and its documentation is displayed: 'write.csv(...) write.table prints its required argument x (after converting it to a data frame if it is not one nor a matrix) to a file or connection. Press F1 for additional help'.
- Plots Panel:** Displays a histogram titled 'Histogram of iris\$sepal.Length'. The x-axis represents sepal length (ranging from 4 to 8) and the y-axis represents frequency (ranging from 0 to 30).
- Code Editor:** Shows the command `> write.csv(sepal.Length, file =` with an auto-completion menu open for the 'file' argument. The menu lists options: x =, file =, append =, quote =, sep =, eol =, na =, and dec =. The 'file =' option is selected, and its documentation is displayed: 'file either a character string naming a file or a connection open for writing. "" indicates output to the console. Press F1 for additional help'.

Essential Data Skills



Connect to MySQL in DSVD

- **Establish connection**

```
> library(DBI)
> library(RMySQL)
> mysqlconnection = dbConnect(MySQL(), user = 'd2dtraining_user',
password= , dbname='GSA_D2D_Training', host
='dc1dbeywj57czbx.c1q8kedtqajq.us-east-1.rds.amazonaws.com')
```

- **Read tables**

```
> dbListTables(mysqlconnection)
[1] "pseudo_facebook"      "reddit"
[3] "statesDataForIntermediateR"
```

- **Create R object**

```
> statesData = dbSendQuery(mysqlconnection, "select * from
statesDataForIntermediateR")
```

- **Convert object to data frame**

```
> dataFrame = fetch(statesData)
> print(dataFrame)s, or
> View(dataFrame)
```

Will be disabled after training
Replace with your own credentials / schema name

Read, Dim, Write

- **Read in data**

```
> statesInfo = read.csv("C:\\Users\\mashk\\Desktop\\R  
Training\\statesDataForIntermediateR.csv")
```

- **Assess the data**

```
> dim(statesInfo)
```

```
[1] 50  12
```

```
> class(statesInfo)
```

```
[1] „data.frame“
```

- **Write data**

```
> write.csv(statesInfo, file = „statesInfo.csv“)
```

- **Find your file**

```
> getwd()
```

Replace “\”
with “/” or “\\”

Factors and Factor Levels

- **str()** – compactly display the internal **structure** of an R object
 - **Factor** – a (categorical) variable with a discrete set of values
 - **Level** - a value a qualitative (categorical) variable can take

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 .
```


Subset Data Frame

- **Select rows with []**

```
> iris[1:5, ]
```

- **Select columns with []**

```
iris[, c("Sepal.Length", "Sepal.Width")]
```

- **subset() function**

```
StatesRegion1 = subset(statesInfo, state.region == 1)
```

```
diamondsSelected = subset(diamonds, cut == "Ideal" & color == "E")
```

```
dsPrice2 = subset(diamonds, cut == "Ideal" & color == "E", c("price", "clarity" ))
```

```
irisPetalData = subset(iris, select = -c(1,2))
```

- Can specify Column after ",",
- Use c() for multiple Columns
- All Columns if empty

- Drop Columns by number

Descriptive Statistics

- R provides a wide range of functions for obtaining summary statistics.
- Built-in functions `sd()`, `mean()`, `median()`, `max/min()`, e.g.:

➤ `mean(x, trim = 0, na.rm = FALSE)`

na – not available
rm - remove

```
> mean(iris$Sepal.Length)
```

```
[1] 5.843333
```

```
> mean(iris$Sepal.Length, trim = .05)
```

```
[1] 5.820588
```

- **summary()**

➤ `summary(StatesRegion1)`

➤ **Quartiles:**

- **1stQu** or lower quartile, is the value that cuts off the first 25% of the data when it is sorted in ascending order. i.e. 70.55 life.exp means that 25% live less than 70.55 years.
- second quartile, or **Median**, is the value that cuts off the first 50%
- **3rdQu**, or upper quartile, is the value that cuts off the first 75%

life.exp	murder	highSchoolGrad	frost	area
Min. :70.39	Min. : 2.400	Min. :46.40	Min. : 82.0	Min. : 1049
1st Qu.:70.55	1st Qu.: 3.100	1st Qu.:52.50	1st Qu.:115.0	1st Qu.: 7521
Median :71.23	Median : 3.300	Median :54.70	Median :127.0	Median : 9027
Mean :71.26	Mean : 4.722	Mean :53.97	Mean :132.8	Mean :18141
3rd Qu.:71.83	3rd Qu.: 5.500	3rd Qu.:57.10	3rd Qu.:161.0	3rd Qu.:30920
Max. :72.48	Max. :10.900	Max. :58.50	Max. :174.0	Max. :47831

Covariances and Correlations

- **Same syntax for both**
- **Correlation between x and y**

`cor(x,y), cov(x,y)`

If x is a matrix, then y is NULL and syntax is `cor(x)`

- **Optional parameters**

➤ **Use**

A character string giving a method for computing in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs"

- **Correlation with missing data**

`cor(x,y, use = "complete")`

➤ **Method**

A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman":

```
> cor(iris$Sepal.Length, iris$Petal.Length)
[1] 0.8717538
> cor(iris$Sepal.Length, iris$Petal.Length, method = "kendall")
[1] 0.7185159
> cor(iris$Sepal.Length, iris$Petal.Length, method = "spearman")
[1] 0.8818981
> cov(iris$Sepal.Length, iris$Petal.Length, method = "spearman")
[1] 1661.304
```

Correlation Test

- Can be based on Pearson, Kendall or Spearman
- Returns correlation coefficient and p-value (significance level)

```
> cor.test(iris$Sepal.Length, iris$Petal.Length)
```

Pearson's product-moment correlation

data: iris\$Sepal.Length and iris\$Petal.Length

t = 21.646, df = 148, p-value < 0.000000000000000022

alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval:

0.8270363 0.9055080

sample estimates:

cor

0.8717538

Pearson Vs Kendal and Spearman

- **Most commonly used is Pearson method**
- **Pearson correlation measures linear dependence between two normally distributed variables when change in one variable causes proportional change in another**
- **Kendal and Pearson are rank-based correlation coefficients, i.e. the data point are first ranked and the correlation of the ranks is measured**
- **Kendal and Pearson can be used for ordinal data, e.g. correlation between test scores and months of training**

ggplot2

- **“ggplot2 is the most elegant and aesthetically pleasing graphics framework available in R. It has a nicely planned structure to it.”**

<http://r-statistics.co/ggplot2-Tutorial-With-R.html>

- **ggplot2 works with data frames and not individual vectors**
- **You can keep enhancing the plot by adding more layers (and themes) to an existing plot created using the ggplot() function**

ggplot2 Cont.

- **Setup**
`options(scipen=999) # turn off scientific notation like 1e+06`
`library(ggplot2)`
`data("midwest", package = "ggplot2")`
- **Initiate**
`ggplot(midwest, aes(x=area, y=poptotal))`
- **Scatter Plot**
`ggplot(midwest, aes(x=area, y=poptotal)) + geom_point()`
- **Smoothing layer**
`g <- ggplot(midwest, aes(x=area, y=poptotal)) + geom_point() +`
`geom_smooth(method="lm")`
`plot(g)`
- **Adjust axis by deleting out of range points**
`g + xlim(c(0, 0.1)) + ylim(c(0, 1000000))`
- **Change labels**
`g1 <- g + coord_cartesian(xlim=c(0,0.1), ylim=c(0, 1000000))`
`g1 + labs(title="Area Vs Population", subtitle="From midwest dataset",`
`y="Population", x="Area", caption="Midwest Demographics")`

ggplot2 Cont. 2

- **Change color and size of points**

```
ggplot(midwest, aes(x=area, y=poptotal)) +  
  geom_point(col="steelblue", size=3) + # Set static color and size for  
  points  
  geom_smooth(method="lm", col="firebrick") + # change the color of line  
  coord_cartesian(xlim=c(0, 0.1), ylim=c(0, 1000000)) +  
  labs(title="Area Vs Population", subtitle="From midwest dataset",  
        y="Population", x="Area", caption="Midwest Demographics")
```

- **Change the Color To Reflect Categories in Another Column**

```
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +  
  geom_point(aes(col=state), size=3) +  
  geom_smooth(method="lm", col="firebrick", size=2) +  
  coord_cartesian(xlim=c(0, 0.1), ylim=c(0, 1000000)) +  
  labs(title="Area Vs Population", subtitle="From midwest dataset",  
        y="Population", x="Area", caption="Midwest Demographics")  
plot(gg)
```


Q & A