LINUX AND R COMPUTATIONAL PROTOCOL: APPLICATION OF STATISTICS AND DATA SCIENCES IN CANCER

CREATED BY
ARLI ADITYA PARIKESIT
NANDA RIZQIA PRADANA RATNASARI
DAVID AGUSTRIAWAN

COURTESSY OF
DEPARTMENT OF BIOINFORMATICS
INDONESIA INTERNATIONAL INSTITUTE FOR LIFE SCIENCES
JUNE 2021

Table of Contents

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Arli Aditya Parikesit
Email : arli.parikesit@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   a. If you have any troubleshoot please contact your supervisor or Building Management
   b. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   c. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   d. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 1

Date : September 3rd, 2019

Laboratory :Bioinformatics laboratory

Overview

- Ubuntu Linux 16.04, which is the latest long-term support version of Linux, is the most common Linux that is simple to use, has a lot of graphical tools, connected to huge online community.
- Terminator, Guake and Clipit are features in Linux that can help users to working effectively. Terminator is another form/version of Linux Terminal that is more convenient to use. When it comes to working long hours in a terminal, Terminator can help users to work fast and easily split screen functionally. Then, Guake is another terminal that opens really fast and whenever users are. Meanwhile, Clipit is a feature for doing copy-paste effectively.

Objective

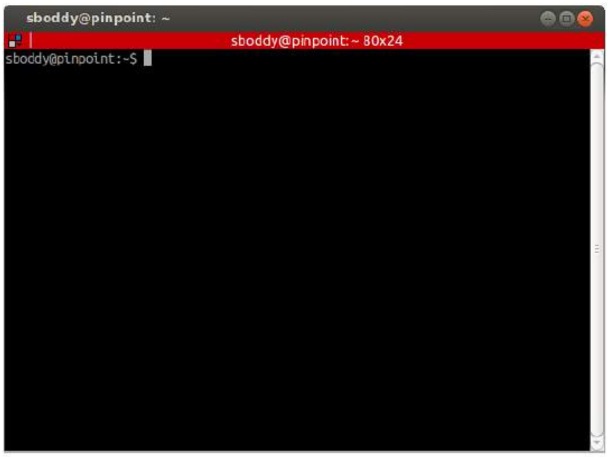
Understand the function of Terminator, Guake and Clipit and able to operate the features.

Procedure

1. Downloading all the sources files in GitHub repository (https://github.com/petruisfan/linux-for-developers)
2. Installing and exploring menus in Terminator.

TERMINATOR

Terminator is a terminal emulator like xterm, gnom-terminal, konsole, etc. The Terminator Application is written and distributed under the terms of the GNU GPL v2 licence. The main Terminator icon was created by Cory Kontros, and provided under the CC-BY-SA licence.

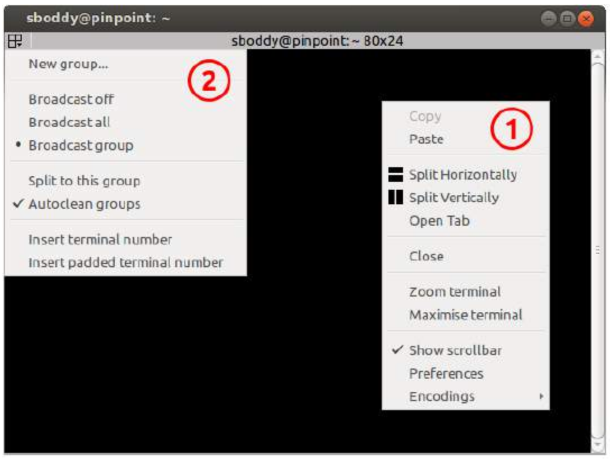


```
sboddy@pinpoint:~$ 
```

Graph 1. Default window for Terminator.

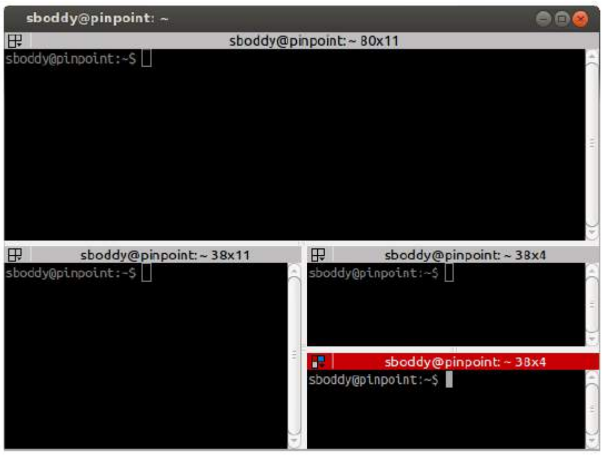
There are two kinds of Menu bar in Terminal, they are:

1. The Context Menu
   This is the main menu reached with right-click over a terminal, and will let you access all the settings, profiles, shortcuts and configurations. It is however kept brief to avoid the mega-menus that sometimes grow unchecked.
2. The Grouping Menu
   This is reached with a click on the trio of coloured boxes in the titlebar. Later, when we cover Grouping and broadcasting to multiple terminals we will cover this properly. For now it is enough to know where it is and how to trigger it.



Graph 2. Two kinds of Menu in Terminators
(You will never see a window that looks like this, as it is impossible to have both menus up at the same time)

The Context Menu
There are five parts of the context menus which are:
- **The standard Copy and Paste** for text that has been highlighted with the mouse. Shortcut for *Copy* is *Shift*+*Ctrl*+C and shortcut for *Paste* is *Shift*+*Ctrl*+V.
- **Split Horizontally and Split Vertically** are used to divide the current space for the current terminal half.
  - *Ctrl* + *Shift* + O to split the screen horizontally
  - *Ctrl* + *Shift* + E to split the screen vertically
  - *Ctrl* + *Shift* + T to create or open a new tab

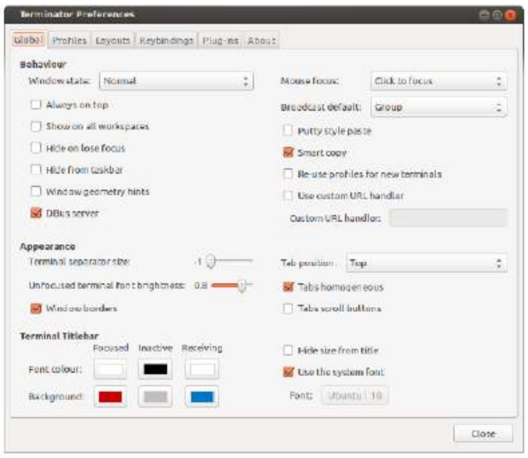


```
sboddy@pinpoint: ~
sboddy@pinpoint:~ 80x11
sboddy@pinpoint:~$

sboddy@pinpoint:~ 38x11
sboddy@pinpoint:~$

sboddy@pinpoint:~ 38x4
sboddy@pinpoint:~$

sboddy@pinpoint:~ 38x4
sboddy@pinpoint:~$
```

Graph 3. Terminal window that has been divided

- **Close** menu. Shortcut for this menu is *Ctrl* + *Shift* + W or *Ctrl* + D.
- **Zoom terminal** will zoom into the current terminal hiding all other terminals and tabs, and increasing the the size of the font. There are some shortcuts in this menus:
  - *Ctrl* + *Shift* + *Arrow*   to resize the screen
  - *Ctrl* + + or *Ctrl* + -   to zoom in or to zoom out the screen
- The fifth part of the menu has three items, which are:
  - **Show scrollbar** will toggle the scrollbar on a per terminal basis. There is also a way to define this in the Profiles.
  - **Preferences** lets users configure and tune Terminator to better suit the needs
  - **Encodings** will allow user to select a different encoding to the default of UTF-8.

Preferences Windows
Global



Behaviour
Window state: Normal
Mouse focus: Click to focus
☐ Always on top
Broadcast default: Group
☐ Show on all workspaces
☐ Putty style paste
☐ Hide on lose focus
✓ Smart copy
☐ Hide from taskbar
☐ Re-use profiles for new terminals
☐ Window geometry hints
☐ Use custom URL handler
✓ DBus server
Custom URL handler:

Appearance
Terminal separator size: 1
Tab position: Top
Unfocused terminal font brightness: 0.8
✓ Tabs homogeneous
✓ Window borders
☐ Tabs scroll buttons

Terminal Titlebar

	Focused	Inactive	Receiving
Font colour:	⬜	⬛	⬜
Background:	🟥	▨	🟦

☐ Hide size from title
✓ Use the system font
Font: Ubuntu 10

Graph 4. Global – Preferences Window

Behaviour

- Window state (default: Normal)
  This will determine what happens on startup normally.
  • Normal - Window opens as normal.
  • Hidden -Window does not open. Useful at login, so it is already available with a shortcut.

- Maximised - Window opens maximised in the standard window manager frame.
- Fullscreen - Window opens fullscreen with no window manager frame.

- Always on top (default: off)
  Window attempts to remain on top.
- Show on all workspaces (default: off)
  The focused window will follow if you switch to a different virtual desktop.
- Hide on lose focus (default: off)
  This is a quake console like feature, where the user want the window to vanish when clicking elsewhere.
  This is rather buggy at the moment as it is very easy for the main window to lose focus and disappear.
- Hide from taskbar (default: off)
  The first window opened will not be displayed in the taskbar. Subsequent windows will show in the taskbar (bug?).
- Window geometry hints (default: off)
  If this is checked, then when resizing Terminator will attempt to step the sizing by the current font, and display a small box with the dimension of the window in characters.
- DBus server (default: on)
  If a Terminator DBus server is not already on the session bus, try to start one.
- Mouse focus (default: Click to focus)
  By what method the mouse pointer sets the focus on a terminal.
  - GNOME Default - Act as per the system settings.
  - Click to focus - You must click with in a terminal to make it the focus.
  - Follow mouse pointer - Moving the pointer over a terminal makes it the focus.
- Broadcast default (default: Group)
  Which broadcast mode should be selected at startup:
  - All - All terminals receive keystrokes.
  - Group - Only terminals in the same group as the current terminal receive keystrokes.
  - None - Only the current terminal receives keystrokes.
- PuTTY style paste (default: off)
  Make the right mouse button operate like in PuTTY, so right-click will paste the Primary selection, and middle-click will open the Context Menu. (For ex-PuTTY users).
- Smart copy (default: on)
  If enabled and there is no selection, the shortcut is allowed to pass through. This is useful for overloading Ctrl+C to copy a selection, or send the SIGINT to the current process if there is no selection. If not enabled the shortcut does not pass through at all, and the SIGINT does not get sent.
- Re-use profiles for new terminals (default: off)

Appearance

- Terminator seperator size (default: -1)
  This is the width in pixels, and can range from -1 to 5. The value of -1 will take the default size from the system theme.
- Unfocused terminal font brightness (default: 0.8)
  Terminals that do not currently have the focus will can be dimmed to aid focus. The value can range from 0 (invisible) to 1 (full brightness)

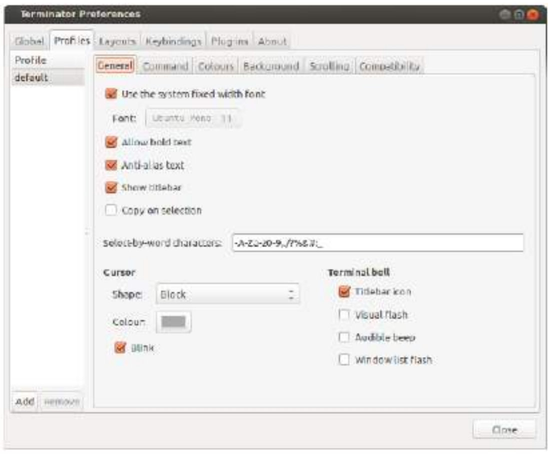
- Window borders (default: on)
  The window manager frame can be removed from your windows.
- Tab position (default: Top)
  Where the tabs will be located within the window
  • Top
  • Bottom
  • Left
  • Right
  • Hidden - Tabs still work, you just can’t see them.
- Tabs homogeneous (default: on)
  Tabs will have equal widths
- Tabs scroll buttons (default: off)
  When there are more tabs than can fit within the window buttons will be drawn for moving left and right.

Terminal Titlebar

There is a table of the colours for the titlebars on the left. These are modelled on those used in a utility . The three sets (Focused, Inactive and Receiving) will make more sense after reading the section about The Grouping Menu.

- Hide size from title (default: off)
  At the end of the label in the titlebar the size of the terminal is given in characters, i.e. (80x24).
  Enabling this item will disable the size text.
- Use the system font (default: on)
  By default the system defined proportional font will be used for the text in the titlebar. Turning
  this off allows you to use a custom font.
- Font (default: inactive, system proportional font)
  If active and set, then the custom font to be used in the titlebar.

Profile
General



[✓] Use the system fixed width font
Font: Ubuntu Mono 11
[✓] Allow bold text
[✓] Anti-alias text
[✓] Show titlebar
[ ] Copy on selection
Select-by-word characters: -A-Za-z0-9,./?%&#:_

Cursor
Shape: Block
Colour: ▨
[✓] Blink

Terminal bell
[✓] Titlebar icon
[ ] Visual flash
[ ] Audible beep
[ ] Window list flash

Graph 5. Profile - Preference Window

- Use the system fixed width font (default: on)
  By default the system defined proportional font will be used for the text in the terminal. Turning this off allows you to use a custom font.
- Font (inactive, system fixed width font)
  If active and set, then the custom font to be used in the terminal.
- Allow bold text (default: on)
  Allows you to disable the use of bold fonts in the terminal.
- Anti-alias text (default: on) †Not in GNOME Terminal
  In Terminator you can turn the font smoothing off. This is no longer possible in GNOME Terminator.
- Show titlebar (default: on)
  The titlebar strip across the top of each terminal can be turned off.
- Copy on selection (default: off)
  This puts the selection into the copy/paste buffer, as well as being available on middle-click.
- Select-by-word characters (default: -A-Za-z0-9,./?%&#:_)
  Using double-click to select text will use this pattern to define what characters are considered part of the word.

Cursor

- Shape (default: Block)
  Set the cursor shape
  • Block - Solid rectangle.
  • Underline - Single pixel tall horizontal line.
  • I-Beam - Single pixel wide vertical line.
- Colour (default: #AAAAAA)
  The colour of the cursor.
- Blink (default: on)
  Whether the cursor blinks on and off.

Terminal bell

- Titlebar icon (default: on)

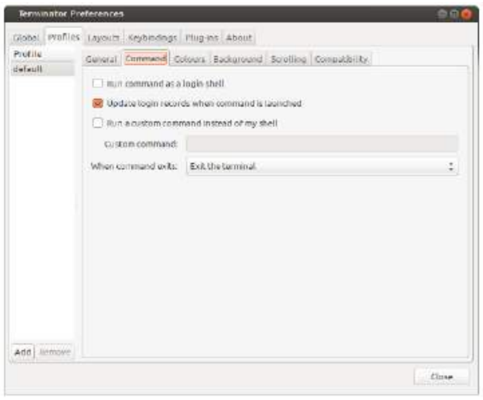
On the right side of the titlebar a small light-bulb icon will be displayed for a few seconds.

- Visual flash (default: off)
  The terminal area will briefly flash.
- Audible beep (default: off)
  The normal system beep noise as defined in system settings.
- Window list flash (default: off)
  This will set the urgent flag on the window in the taskbar. The actual effect will be taskbar dependant.

Not in Terminator

- Profile name
  Our profiles names are in the sidebar to the left.
- Show menubar by default in new terminals
  Terminator doesn't use a traditional menu bar.
- Terminal bell
  Terminator has more options, so has four separate options in their own grouping. This item in GNOME Terminal is the same as Audible beep defined above.
- Use custom default terminal size
  Terminator handles window sizes within Layouts, or with Command line options.

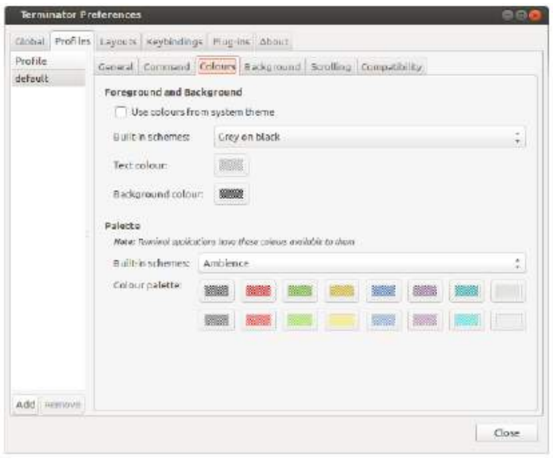
Command



[ ] Run command as a login shell
[✓] Update login records when command is launched
[ ] Run a custom command instead of my shell
Custom command:
When command exits: Exit the terminal

- Run commands as a login shell (default: off)
  Force the command to run as a login shell.
- Update login records when command is launched (default: on)
  Updates login records when a new shell is opened.
- Run a custom command instead of my shell (default: off)
  Enable the use of a custom command instead of the users default shell.
- Custom command (default: inactive, empty)
  If enabled and set, the users default shell will be replaced with the command specified here.
- When command exits (default: Exit the terminal)
  When the running command exits (default or custom) what action should be taken.
  • Exit the terminal - Terminal closes, causing layout to adjust.
  • Restart the command - Original command restarts immediately.
  • Hold the terminal open - The terminal and scrollback will remain visible and accessible until the user explicitly closes the terminal, or closes the window.

Color


Terminator Preferences

Profile: default

Foreground and Background
☐ Use colours from system theme
Built-in schemes: Grey on black
Text colour:
Background colour:

Palette
Note: Terminal applications have these colours available to them
Built-in schemes: Ambience
Colour palette:

Foreground and Background

- Use colours from system theme (default: off)
  Use colours as defined in the system theme. Not clear at this time where exactly these come from. Differences in the GTK2, GTK3 and GNOME Terminal.
- Built-in schemes (default: Grey on black)
  Pick a primary colour combination for foreground and background. Again there are unexplained differences between Terminator and GNOME Terminal.
  The list seems to be dynamic and vary depending on the system, with the addition of Custom which allows setting the colours as desired.
- Text colour (default: inactive, #AAAAAA)
  If the Built-in schemes is set to Custom the text colour can be set here.
- Background colour (default: inactive, #000000)
  If the Built-in schemes is set to Custom the background colour can be set here.

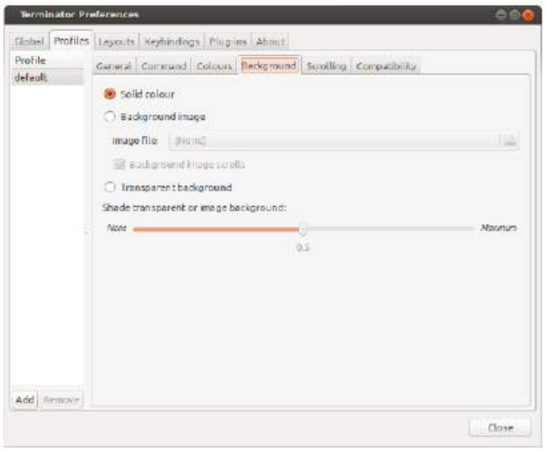
Palette

- Built-in schemes (default: Ambience)
  A predefined colour palette can be selected. Again there are unexplained differences between
  Terminator and GNOME Terminal.
  The default here may be system dependant, with Ambience being an Ubuntu colour scheme.
- Colour palette (default: inactive)
  If the Palette's Built-in schemes is set to custom, a set of colour swatches are used to configure the 16 primary colours of the shell palette.

Not in Terminator

- Bold colour
  In theory nothing is stopping us implementing this, it just doesn't appear to have ever been added.
- Same as text colour
  In truth, I'm not exactly sure what this does, but at a guess, the user can force bold to be drawn in the same colour as the foreground text.

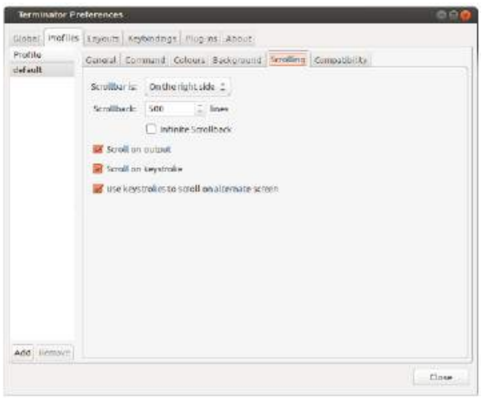
Background



(●) Solid colour
( ) Background image
Image file: (None)
[✓] Background image scrolls
( ) Transparent background
Shade transparent or image background: None — 0.5 — Maximum

- Solid colour (default: active)
  Background of terminal is set to the solid colour set in previous Colours tab.
- Background image (default: inactive)
  Background will be an image. There is no scaling done.
- Image file (default: inactive, None)
  If Background image is set, then the image to use can be selected here.
- Background image scrolls (default: inactive, on)
  If the Background image is set, then setting this to on will cause the background image to change as the window moves. This is a for of fake transparency.
- Transparent background (default: inactive)
  This will attempt true transparency where the windows below are partially visible through the terminal.
- Shade transparent or image background (default: 0.5)
  For Background image and Transparent background this is how much the solid colour should be blended in, giving a tinting effect.

Scrolling



Scrollbar is: On the right side
Scrollback: 500 lines
[ ] Infinite Scrollback
[✓] Scroll on output
[✓] Scroll on keystroke
[✓] Use keystrokes to scroll on alternate screen

- Scrollbar is (default: On the right side)
  If and where the scrollbar should appear.
  • On the left side
  • On the right side
  • Disabled
- Scrollback (default: 500 lines)

How many lines to keep before discarding.

- Infinite Scrollback (default: off)
  Lines are never discarded, and all lines since the session began are available.
  Note: Data is placed onto the disk by the underlying VTE component, so even after a long time, the
  memory footprint and performance of Terminator should be OK.
- Scroll on output (default: on)
  Moves terminal to end of scrollback buffer when any output occurs.
- Scroll on keystroke (default: on)
  Moves terminal to end of scrollback buffer when any keypress occurs.
- Use keystrokes to scroll on alternate screen (default: on)

3. Installing and exploring menus in **Guake**.
   a. open the terminal and type sudo apt install guake
   b. open the Guake terminal by pressing F12
   c. go to **shell** and click **open new tab in current directory** button
   d. go to **scrolling** and insert a big number in **scrollback lines** to ensure that users have enough space for writting the instructions/codes
   e. go to sub-menu **appearance** and explore/change text color, background color and cursor blink mode.
   f. users can resize the Guake by dragging the margin or make it full screen by pressing F11.
   g. Guake does not start automatically when Ubuntu reboot, so users need to add it to a startup application.

4. Installing and exploring menus in Clipit.
   a. open the terminal and type sudo apt install clipit to install it
   b. to call ClipIt, hit *Ctrl*+*Alt*+H
   c. for example:
      - open a file in terminal
      - then source a text or variable that will be copied
      - hit *Ctrl*+*Alt*+H on a screen
      - a variable/text that has been sourced will appear on the screen then click it.

Discussion.

1. What is the goals of Terminator development project purposed by its creator?
2. Mention some programs that inspired the creator to establish Terminator!
3. Name a terminal that become the basis of Terminator behaviours!
4. Mention some features in Terminators!
5. Terminator is an application written and distributed under the term of a licence. What licence is it?

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : David Agustriawan
Email : arli.parikesit@i3l.ac.id

Supervisor(s)	email(s)
Arli Aditya Parikesit	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   e. If you have any troubleshoot please contact your supervisor or Building Management
   f. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   g. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   h. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 2

Date : September 10td, 2019

Laboratory :Bioinformatics laboratory

Overview

- Vim is a commonline interface and a standarlone application in a graphical user interface. It is an attractive and flexible tool for controlling text editing environment. Vim provides basic commands to work with CRUD (create, read, update and delete) operations. User, also, can upgrade the application from shell intepreter to zsh framework. Tha application assign grep comment for some basic regular expressions and subshells to run embedded commands.

Objective

Students are expected to understand and able to work with Vim, able to manage zsh using the `oh-my-zsh` framework, able to write and run super powerful one line commands using pipes and able to explore the shell scripting libraries.

Procedure

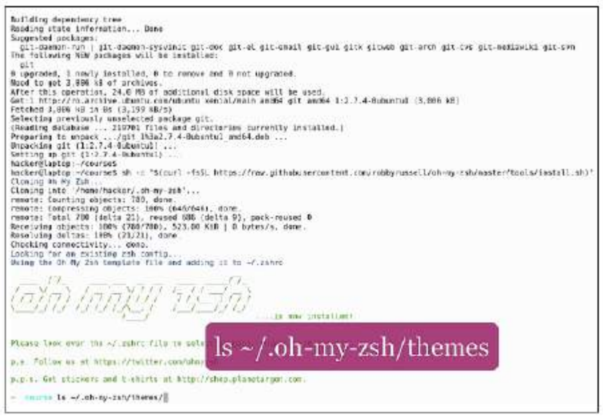
Working with Vim

- Open the terminator and type sudo apt install vim to install Vim
- To start editing text, press the *Insert* key (this will take us to the insert mode)
- Press the *Insert* key again to go to replace the mode and override text
- Press the *Esc* key to exit insert or replace
- Type *yy* to copy a line
- Type *p* to paste the line
- Type *dd* to cut the line
- Type *:w* to save any changes
- To save the file in editing text, type `vim.txt`
- Type `:q` to exit Vim
- Type `:wq`: Write and exit at the same time
- Type `:q!`: Exit without saving

Oh-my-zsh

- Open the terminator and type sudo apt install zsh to install zsh
- Go to link https://github.com/robbyrussell/oh-my-zsh, and follow the instructions for installing the oh-my-zsh framework
  - Type `sudo apt install git`
  - Via Curl **sh -c "$(curl -fsSL https://raw.githubusercontent.com/robbyrussell/oh-myzsh/master/tools/install.sh)"**

   - Via wget **sh -c "$(wget https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools/install.sh -O -)"**
- Pick themes by typing **ls ~/.oh-my-zsh/themes**



```
Building dependency tree
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-arch git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 3,886 kB of archives.
After this operation, 24.0 MB of additional disk space will be used.
Get:1 http://ro.archive.ubuntu.com/ubuntu xenial/main amd64 git amd64 1:2.7.4-0ubuntu1 (3,886 kB)
Fetched 3,886 kB in 0s (3,199 kB/s)
Selecting previously unselected package git.
(Reading database ... 219701 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.7.4-0ubuntu1_amd64.deb ...
Unpacking git (1:2.7.4-0ubuntu1) ...
Setting up git (1:2.7.4-0ubuntu1) ...
hacker@laptop:~/courses
hacker@laptop:~/courses$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"
Cloning Oh My Zsh...
Cloning into '/home/hacker/.oh-my-zsh'...
remote: Counting objects: 780, done.
remote: Compressing objects: 100% (646/646), done.
remote: Total 780 (delta 21), reused 686 (delta 9), pack-reused 0
Receiving objects: 100% (780/780), 523.00 KiB | 0 bytes/s, done.
Resolving deltas: 100% (21/21), done.
Checking connectivity... done.
Looking for an existing zsh config...
Using the Oh My Zsh template file and adding it to ~/.zshrc

                        ....is now installed!

Please look over the ~/.zshrc file to sele

p.s. Follow us at https://twitter.com/ohmy

p.p.s. Get stickers and t-shirts at http://shop.planetargon.com.

➜  courses ls ~/.oh-my-zsh/themes/
```

- To change the theme, type `open~/.zshrc`
- Type `mkdir 'directory_name'` to create directory
- Type `cd 'directory_name'` to change the directory into the new one
- Type `git init` to get started the git
- Create a file by typing `touch 'file_name'`
- Type `git status` to clean and verify directory

You can now see the branch name in the prompt, and there are some other cool features that you might like to explore:

- **Command completion**: Start typing, for example, ip, and press *Tab*. We can see all the commands that start with IP and we can hit *Tab* again to start navigating through the different options. We can use the arrow keys to navigate and hit *Enter* for the desired command
- **Params completion**: For example type `ls -` and press *Tab*, and we can see here all the options and a short description for each. Press *Tab* again to start navigating through them and *Enter* to select.
- **History navigation**: Click on arrow up key to search in history, filtering by the string that is written before the cursor. For example, if I type `vim` and press the arrow up key, I can see all the files opened with Vim in my history.
- **History search**: Press *Ctrl* + *R* and start typing, and press *Ctrl* + *R* again to search the same occurrence in history. For example ~, and *Ctrl* + *R* to see all commands that have ~ in the string.
- **Navigating**: Here press *Ctrl* + arrow left/right to jump one word, *Ctrl* + *W* to delete one word, or *Ctrl* + *U* to delete the whole line.
- **cd completion case insensitive**: For example, `cd doc` will expand into `cd Documents`.
- **cd directory completion**: If you are lazy and want to specify only a few key letters in a path, we can do that too. For example, `cd /us/sh/zs` + *Tab* will expand into `cd /usr/share/zsh`.

- **Kill completion:** Just type `kill` and *Tab* and you will see a list of `pids` to kill. From there you can choose which process to kill.
- **chown completion**: Type `chown` and tab to see a list of users to change owner to. The same applies to groups.
- **Argument expansion**: Type `ls *` and hit *Tab*. You see `*` expanded to all files and folders in the current directory. For a subset, type `ls Do*` and press *Tab*. It will only expand to documents and downloads.
- **Adds lots of aliases:** Just type alias to see a full list. Some very useful ones are:
  ```
  .. - go up one folder
  … - go up two folders
  - - cd o the last directory
  ll - ls with -lh
  ```

Basic Regular Expression

- Getting started by opening the file to see the content
- Splitting the screen to see both the file and command side by side
- The simplest commond regular expression is to find a single word
  - We can use file the `grep "word" file.txt` command
  - `word` is the string we are looking for and `file.txt` is the file where we perform the search
  - You can see that grep printed the line that contained our string and the word is highlighted with another color. This will only match the exact case of the word.
  - To do a case insensitive search, `grep` has an `-i` option. What this means is that grep will print the line that contains our word even if the word is in a different case, by typing `grep -i "joe" file.txt`
  - If we don't know exactly what characters are there in our string, we can use `.*` to match any number of characters. Command (`.`) is to match any character and (`*`) is to match previous character multiple times.
  - A very common scenario is to find empty lines in a file. For this we use the `grep "^\s$" file.txt` command:
    - o Where `\s` : This stands for space
    - o `^` : It's for the beginning of the line
    - o `$` : It's for its ending.
  - `grep` can do a neat little trick to count the number of matches. For this, we use the `-c` parameter
  - To find all the lines that have only letters and space, use:
    - o `grep`
    - o `""`: Open quotes
    - o `^$`: From the beginning of the line to the end
    - o `[]*`: Match these characters any number of times
    - o `A-Za-z`: Any upper and lower case letter
  - If we run the command up to here, we get only the first line. If we add:
    - o - 0-9 any number we match another two lines,

- o And if we add: - \s any space, we also match the empty lines and the all
- o caps line
- o If we run the command until here, we get only the first line from the output,
- o the rest is not displayed
- o Then, if we add 0-9 we match any number (so the first two lines get matched)
- o And if we add \s we match any type of space (so the empty lines are matched as well)

- **`grep "^[A-Za-z0-9\s]*$" file`** is to find something that is not strings.
  - o **`grep "^[^0-9]*$" file.txt`**
    This command will find all the lines that do not have only numeric characters. `[^]` means match all characters that are not inside, in our case, any non-number.
  - o **`grep "\[.*\]" file.txt`**
    This is for any line that has characters in square brackets.
  - o To find all lines that have these character `!`, type this: **`grep "\!" file.txt.`**

<u>**Pipes and Subshells**</u>

- **`pwd | wc -c` to counth** the length of the current path (pwd stands for `print working directory`)
- **`df -h | grep /home | tr -s " " | cut -f 2 -d " "`** to find used space on the drive
  - `"df -h"`: This shows the disk usage in a human-readable format
  - `"| grep /home"`: This shows only the home directory
  - `'| tr -s " "'`: This substitutes multiple spaces for just one space
  - `'| cut -f 2 -d " "'`: This selects the second column using a space as the delimiter
- **ls -p | grep / | wc -l :** To count all the directories in a folder
- Pipes are a great option to find and kill processes. Say we want to find the process ID of `nautilus`, and `kill all` running instances. For this we use:
  `ps aux | grep nautilus | grep -v grep | awk '{print $2}' | xargs kill`
  - `ps aux`: This prints all processes with PID
  - `| grep nautilus`: Find the ones matching nautilus
  - `| grep -v grep`: Inverts `grep` to exclude the `grep` process
  - `| awk '{print $2}'`: Selects the second word in the line, which is the PID
  - `| xargs kill`: Here `xargs` is used to distribute each PID to a kill command. It is especially used for commands that do not read arguments from standard input.
- To get the IP address of a specific interface (in our case the wireless interface `wlp3s0`), run this: **`ifconfig wlp3s0 | grep "inet addr:" | awk '{print $2}' | cut -f 2 -d ":"`**
  - `ifconfig wlp3s0`: Prints networking information for the `wlp3s0` interface

    - `| grep "inet addr:"`: Gets the line with the IP address
    - `| awk '{print $2}'`: Selects the second word in the line (we could have used cut as well)
    - `| cut -f 2 -d ":"`: This is split by `":"`, and only prints the second word
- In order to get the word frequency, use this: **`cat lorem.txt | tr " " "\n" | grep -v "^\s*$" | sed "s/[,.]//g" | sort | uniq -c | sort -n`**
    - `cat lorem.txt`
    - `| tr " " "\n"`: Transforms each space into a new line character
    - `| grep -v "^\s*$"`: Eliminates empty lines
    - `| sed "s/[,.]//g"`: Eliminates commas (,) and periods (.) to select only the words
    - `| sort`: Sort the results alphabetically
    - `| uniq -c`: Show only unique lines
    - `| sort -n`: Sorts by numerical value
- **`$(ls)`**

  The `ls` subshell returns the files and folders in the current directory and the `ls` from
  outside the subshell will list those individually, showing additional details:
    - Counting all files and directories in the current directory
    - Given the fact that commas (,) and periods (.) are hard links that mark the current and parent directory, we need to count all entries minus these two
    - This can be done using the `expr $(ls -a | wc -l ) - 2` command

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Arli Aditya Parikesit
Email : arli.parikesit@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   i. If you have any troubleshoot please contact your supervisor or Building Management
   j. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   k. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   l. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 3

Date : September 17td, 2019
Laboratory :Bioinformatics laboratory

Overview

Vim is a commonline interface and a standarlone application in a graphical user interface. It is an attractive and flexible tool for controlling text editing environment. Vim provides basic commands to work with CRUD (create, read, update and delete) operations. User, also, can upgrade the application from shell intepreter to zsh framework. Vim can come in handy with its option of encrypting files for storing your passwords.

Objective

Students are expected to understand on how applying Linux development tools.

Procedure

Let's start by opening a new hidden file name.vimrc in our home folder and pasting a few lines:
Typing these code on terminal

```
set nocompatible
filetype off

" Settings to replace tab. Use :retab for replacing tab in existing
files.
set tabstop=4
set shiftwidth=4
set expandtab

" Have Vim jump to the last position when reopening a file
if has("autocmd")
      au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") |
      exe "normal! g'\"" | endif
endif

" Other general vim options:
syntax on
set showmatch " Show matching brackets.
set ignorecase " Do case insensitive matching
set incsearch " show partial matches for a search phrase
set nopaste
set number
set undolevels=1000
```

- `set nocompatible` : means that the Vim mode is not compatible for the old Vim. It is the recommended mode.

- The lines starting with ” are comments, so they can be ignored.
- `tabstop, shiftwidth, expandtab` are comments to use *space* instead of *tabs*
- the `if` and `endif` function is to always open a file and set the cursor in the same position as the last time the file was opened
- `syntax on`: This enables syntax highlighting, so it is easier to read code
- `set nopaste`: This sets nopaste mode, which means you can paste code without having Vim try to guess how to format it
- `set number`: This tells Vim to always show the line numbers
- `set undolevels=1000`: This tells Vim to remember the last 1000 changes we made to the file, so that we can easily undo and redo
- `set showmatch` : to show matching brackets
- `set ignorcase` : to do case insensitive matching
- `set incsearch` : to show partial matches for a search phrase

```
" Always show the status line
set laststatus=2

" Format the status line
set statusline=\ %{HasPaste()}%F%m%r%h\ %w\ \ CWD:\ %r%{getcwd()}%h\ \
\ Line:\ %l\ \ Column:\ %c

" Returns true if paste mode is enabled
function! Has Paste()
       if &paste
       return 'PASTE MODE '
       en
       return ''
end function
```

- `set laststatus` : to show the status line
- `set statusline` : to format status line (contains the name of the file, the current directory, the line and column number)

Vim also has the option of changing the color scheme. To do this, go to `/usr/share/vim/vim74/colors` and choose a color scheme from there.

- `:colorscheme desert` : to set the color scheme

Indentation

Indentation can be done by:

- going into visual model of Vim
- typing V for selecting portions of text or V for selecting full lines
- then pressing > for right indentation or < for left indentation
- repeat . to repeat the last operation

Undo and Redo

- any operation can be undone by hitting `Ctrl+U`
- the operation can be redone by hitting `Ctrl+R`

Changing the case of lettes

- hitting `U` (kapital letter) to make all text upper case
- hitting `u` (lower case) to make all text lower case
- hitting ~ to reverse the current case

Other handy shortcase

- `G`: Go to end of file
- `gg`: Go to start of file
- `Select all`: This is not really a shortcut, but a combination of commands: `gg V G`, as in go to start of file, select full line, and move to the end.
- Hitting K for opening man pages for the word under the cursor
- Typing / + the text to find + Enter : for searching a specific text
- Hit `n` for next occurrence, *N* for previous occurrence

```
:1,$s/CWD/DIR/g
:1,$ - start from line one, till the end of the file
s - substitute
/CWD/DIR/ - replace CWD with DIR
g - global, replace all occurrences.
```

The codes in the box show the way to replace all occurrences of the string `CWD` with the string `DIR`

- **`:10,20s/^/#\ /g`** : is used to comment out lines 10 to 20 in a shell script
- **`:30,$d`** : is for deleting lines of text (deleting everything from line 30 to the end)
- To edit the command we just wrote and run it again, we can open the command history by hitting *`q:`*, navigate to the line containing the command to edit, press Insert, update the line, and press *`Esc`* and *`Enter`* to run the command

Sorting

- Let's create a file with unsorted lines of text from the classic lorem ipsum text:
  `cat lorem.txt | tr " " "\n" | grep -v "^\s*$" | sed "s/[,.]//g" > sort.txt`

- Open `sort.txt` and `run :sort`. We see that the lines are all sorted alphabetically.

Spliting the screen for editing text

- Typing `split`: for horizontal split
- Typing `vsplit`: for vertical split
- `Ctrl+W` : to switch between the splitted windows
- Hiting `e` to open another file in the splitted pane
- To close the window, press :`q`
- Typing `Tab` to write filenames
- Hitting `:w`, followed by the new file name to save the file with different name (save as)

- **`vim file1 file2 file3`** to open multiple files
- After multiple files are open, use `:bn` to move to the next file. To close all the files, hit `:qa`
- open Vim and hit `:Explore` to navigate through the directory layout and open new files
- `vimdiff` : to exit and open two files

Plugin Steroids for Vim

- To manage plugins, let's install the plugin manager pathogen. Open: https://github.com/tpope/vimpathogen.
- Follow the installation instructions. As you can see, it's a one-line command: **`mkdir -p ~/.vim/autoload ~/.vim/bundle && \curl -LSso ~/.vim/autoload/ pathogen.vim https://tpo.pe/pathogen.vim`**
- And after it finishes, add pathogen to your `.vimrc`: `execute pathogen#infect()`
- Open: `https://github.com/scrooloose/nerdtree`, and follow the instructions fo installing it: **`cd ~/.vim/bundle git clone https://github.com/scrooloose/nerdtree.git`**
- Let's open a file and type `:NERDtree`. We see the tree-like structure of our current folder here, where we can browse and open new files.
- Another awesome plugin that comes in really handy is called **Snipmate** and is used for writing code snippets. To install it, go to this link and follow the instructions: https://github.com/garbas/vim-snipmate.
- As we can see, before installing `snipmate`, there is another set of plugins that needs to be installed:
  - `git clone https://github.com/tomtom/tlib_vim.git`
  - `git clone https://github.com/MarcWeber/vim-addon-mw-utils.git`
  - `git clone https://github.com/garbas/vim-snipmate.git`
  - `git clone https://github.com/honza/vim-snippets.git`
- Type `fun` and hit *Tab* for the function autocomplete.
- To write a function name and hit *Tab* to go to the next variable to complete

Vim Password Manager

- Typing `:set cryptmethod?` to see the `cryp` method that Vim is currently using
- Typing `:h 'cryptmethod'` to see different alternatives
- Typing `:set cryptmethod=blowfish2` to change the encryption method
- `vimrc` to create default encryption
- Let's open up a new file, add some dummy passwords inside, and save it. The next step is to encrypt the file with a password, and for this we type `:X`.
- If you exit without saving the file, the encryption will not be applied. Now, encrypt it again, save, and exit the file.

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : David Agustriawan
Email : david.agustriawan@i3l.ac.id

Supervisor(s)	email(s)
Arli Aditya Parikesit	arli.parikesit@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   m. If you have any troubleshoot please contact your supervisor or Building Management
   n. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   o. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   p. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 4

Date : September 24th, 2019

Laboratory :Bioinformatics laboratory

Overview

Tmux is a program that can help users operating hours of refractor turn into a few minutes. Users can run long lasting commands, split screens, and never lose your work with the help of the best terminal multiplexor. It will also help users learning how to discover and interact with your network with the help of commands like netstat and nmap. Autoenv helps switch environments automatically and how to use rm command to interact with trash from command line using the trash utility.

Objective

Students are expected to understand on how applying Linux development tools.

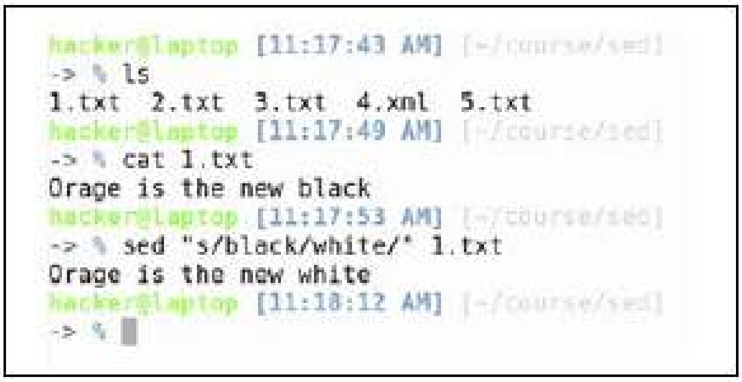
Procedure

Sed

Sed one liner is a tool to help user simplifying thousand lines of codes. It has a cryptical usage but can serve a very powerful tool for quickly editing large amounts of data.

Replacing words

- Firstly, created files that will be edited.
- Then open the file
- Create a simple `sed` command to replace the word *black* to *white*
- Type argument: **`sed "s/black/white/" 1.txt`**
  It is divided into three parts that are: the first part is **`s`** for subtitute; the second part is the word to be replaced which is *black*; the third part is the replacement word which is *white*.

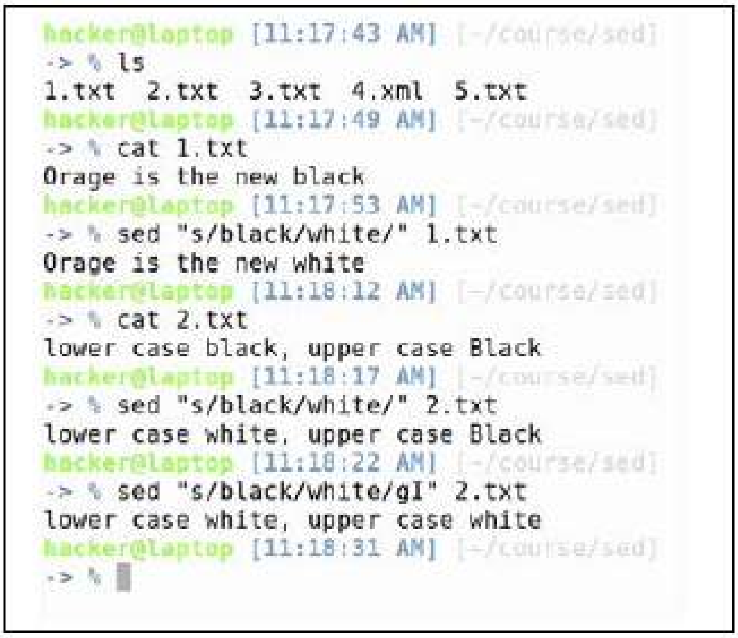


```
hacker@laptop [11:17:43 AM] [~/course/sed]
-> % ls
1.txt  2.txt  3.txt  4.xml  5.txt
hacker@laptop [11:17:49 AM] [~/course/sed]
-> % cat 1.txt
Orage is the new black
hacker@laptop [11:17:53 AM] [~/course/sed]
-> % sed "s/black/white/" 1.txt
Orage is the new white
hacker@laptop [11:18:12 AM] [~/course/sed]
-> %
```

- Now, the result will be printed on the screen, and you can see the word black has been replaced by white.

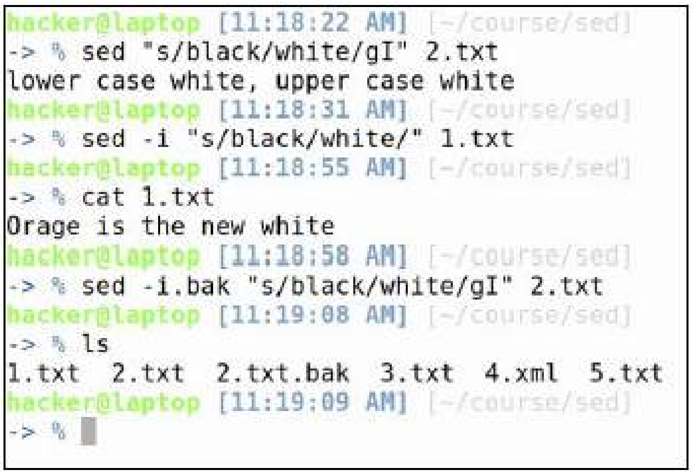
The next case is replacing words that are sensitive to the upper and lower case. Users can add two more characters to the end of the **`sed`** command.

- `g`: It means global replace, used for replacing all the occurrences in the file. Without this, it will only replace the first argument.
- `I`: means case insensitive search.
  `sed "s/black/white/gI" 2.txt`



```
hacker@laptop [11:17:43 AM] [~/course/sed]
-> % ls
1.txt  2.txt  3.txt  4.xml  5.txt
hacker@laptop [11:17:49 AM] [~/course/sed]
-> % cat 1.txt
Orage is the new black
hacker@laptop [11:17:53 AM] [~/course/sed]
-> % sed "s/black/white/" 1.txt
Orage is the new white
hacker@laptop [11:18:12 AM] [~/course/sed]
-> % cat 2.txt
lower case black, upper case Black
hacker@laptop [11:18:17 AM] [~/course/sed]
-> % sed "s/black/white/" 2.txt
lower case white, upper case Black
hacker@laptop [11:18:22 AM] [~/course/sed]
-> % sed "s/black/white/gI" 2.txt
lower case white, upper case white
hacker@laptop [11:18:31 AM] [~/course/sed]
-> %
```

- to save the result in file, use the `-i` argument, which stands for inline replace.
- to backup the result, specify a suffix after `-i` which will create a backup file.
  `sed -i.bak "s/black/white/g" 2.txt`



```
hacker@laptop [11:18:22 AM] [~/course/sed]
-> % sed "s/black/white/gI" 2.txt
lower case white, upper case white
hacker@laptop [11:18:31 AM] [~/course/sed]
-> % sed -i "s/black/white/" 1.txt
hacker@laptop [11:18:55 AM] [~/course/sed]
-> % cat 1.txt
Orage is the new white
hacker@laptop [11:18:58 AM] [~/course/sed]
-> % sed -i.bak "s/black/white/gI" 2.txt
hacker@laptop [11:19:08 AM] [~/course/sed]
-> % ls
1.txt  2.txt  2.txt.bak  3.txt  4.xml  5.txt
hacker@laptop [11:19:09 AM] [~/course/sed]
-> %
```

<u>Create curly brackets</u>:

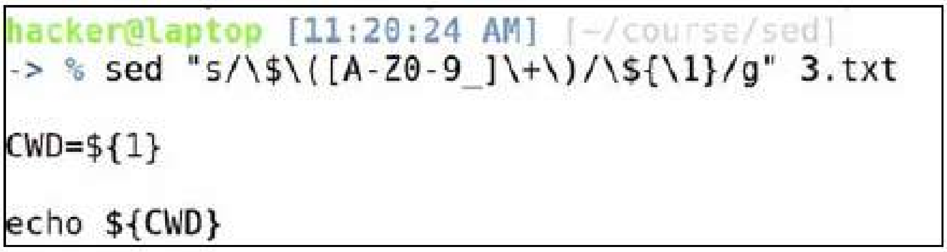
- `s`: It's for substitute.
- `g`: It's for global; meaning replace all occurrences found.
- `\$`: This matches all strings starting with the dollar sign. Here dollar needs to be escaped, so that it's not confused with the *start of the row* anchor.
- We will enclose the string following `$` in ( ), so that we can reference it in the replace part of our command.

- `[ ]`: This is for specifying a range of characters
- `A-Z`: It matches all uppercase characters
- `0-9`: It matches all numbers
- `_`: It matches _
- `\+`: Any character in the [ ] must appear one or multiple times

In the replace part, we will use:

- `\$`: The dollar sign
- `{ }`: The curly brackets we want to add.
- `\1`: The string that was previously matched in the ( )

```
sed 's/\$\([A-Z0-9_]\+\)/\${\1}/g' 3.txt
```

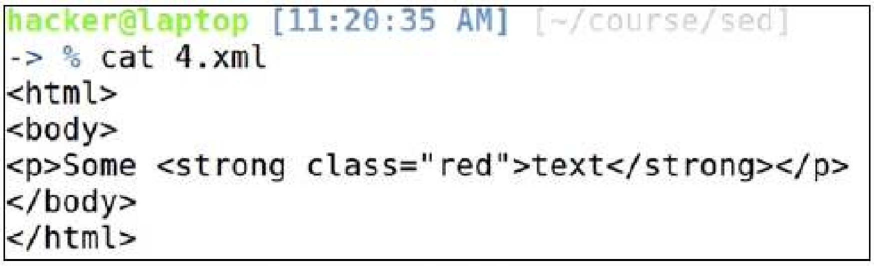


```
hacker@laptop [11:20:24 AM] [~/course/sed]
-> % sed "s/\$\([A-Z0-9_]\+\)/\${\1}/g" 3.txt

CWD=${1}

echo ${CWD}
```

Other common scenarios are replacing content in `xml` or `html` files.
Here we have a basic html file with a `<strong>` text inside. Now, we know that the `<strong>` text has more semantic value for search engine optimizations, so maybe we want to make our strong tags be a simple `<b>` (bold), and manually decide the `<strong>` words in the page. For this we say:

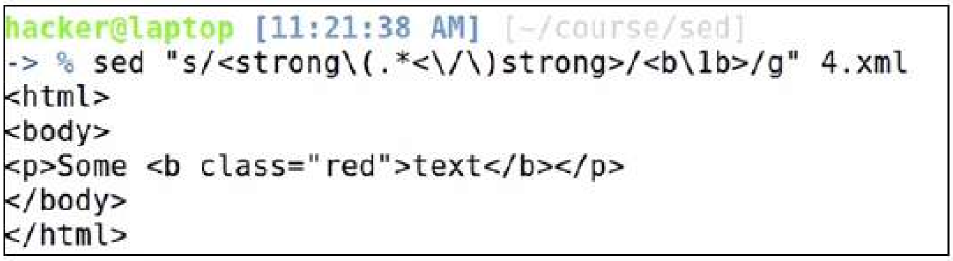
- `s`: This is for substitute.
- `<strong`: The actual text we are searching for.
- `\( \)`: This will be used again for selecting a piece of text, that will be added back.
- `.*`: This means any character, found any number of times. We want to select everything between "`<strong`" and "`strong>`".
- `</`: This is the closing of the tag. This, we want to keep intact.
- `<b\1b>`: Just add `<b b>`, and the text that you previously found in the ( ).

```
sed "s/<strong\(.*</\)strong>/<b\1b>/g" 4.xml
```



```
hacker@laptop [11:20:35 AM] [~/course/sed]
-> % cat 4.xml
<html>
<body>
<p>Some <strong class="red">text</strong></p>
</body>
</html>
```

As you can see, the text was updated correctly, the `red` class still applies to the new tag, and the old text is still contained between our tags, which is exactly what we wanted:



```
hacker@laptop [11:21:38 AM] [~/course/sed]
-> % sed "s/<strong\(.*<\/\)strong>/<b\1b>/g" 4.xml
<html>
<body>
<p>Some <b class="red">text</b></p>
</body>
</html>
```

Deleting (deleting lines of the text)

For example, the file's name is `5.txt` . If we want to delete line 3 from the text, we can type

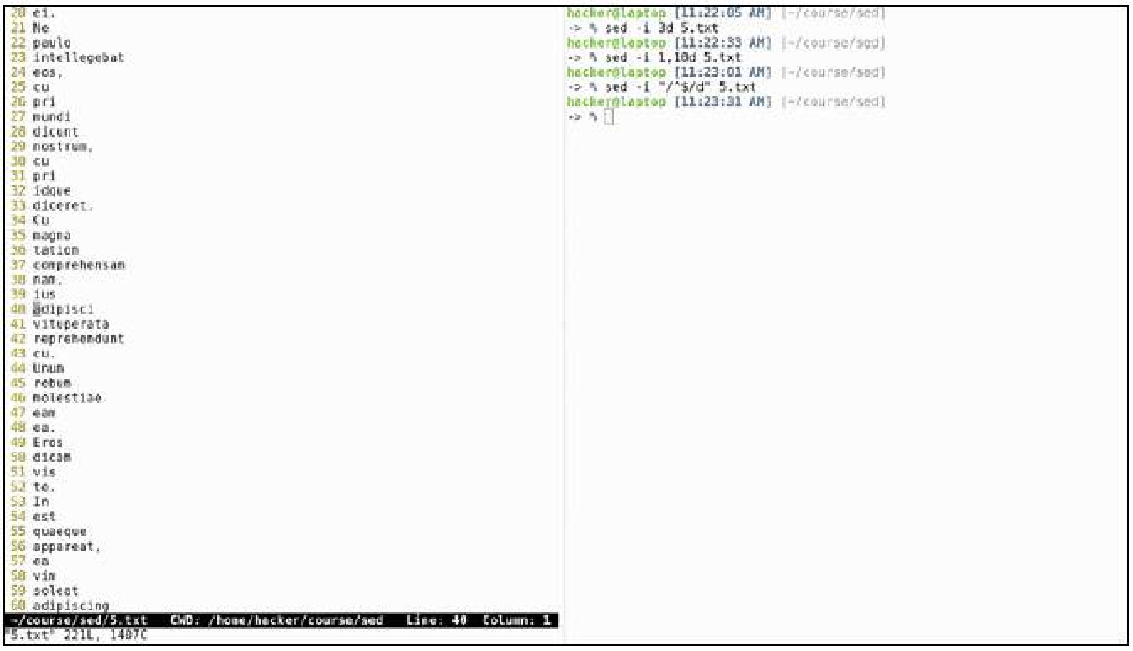
`sed -i 3d 5.txt`

- Hit *:e,* to reload the file in vim, and we see the word `dolor` is no longer there. If, for example, we wanted to delete the first 10 lines of the file, we'd simply run:

`sed -i 1,10d 5.txt`

- Hit *:e*, and you see the lines are no longer there. For our last example, if we scroll down, we can see multiple empty lines of text. These can be deleted with:

  `sed -i "/^$/d" 5.txt`



```
ei.
Ne
paulo
intellegebat
eos,
cu
pri
mundi
dicunt
nostrum,
cu
pri
idque
diceret.
Cu
magna
tation
comprehensam
nam.
ius
adipisci
vituperata
reprehendunt
cu.
Unum
rebum
molestiae
eam
ea.
Eros
dicam
vis
te.
In
est
quaeque
appareat,
ea
vim
soleat
adipiscing
~/course/sed/5.txt   CWD: /home/hacker/course/sed   Line: 40  Column: 1
"5.txt" 221L, 1487C
```

```
hacker@laptop [11:22:05 AM] [~/course/sed]
-> % sed -i 3d 5.txt
hacker@laptop [11:22:33 AM] [~/course/sed]
-> % sed -i 1,10d 5.txt
hacker@laptop [11:23:01 AM] [~/course/sed]
-> % sed -i "/^$/d" 5.txt
hacker@laptop [11:23:31 AM] [~/course/sed]
-> %
```

Which stands for:

- ^: Beginning of line anchor
- $: End of line anchor
- d: Delete

Finding files

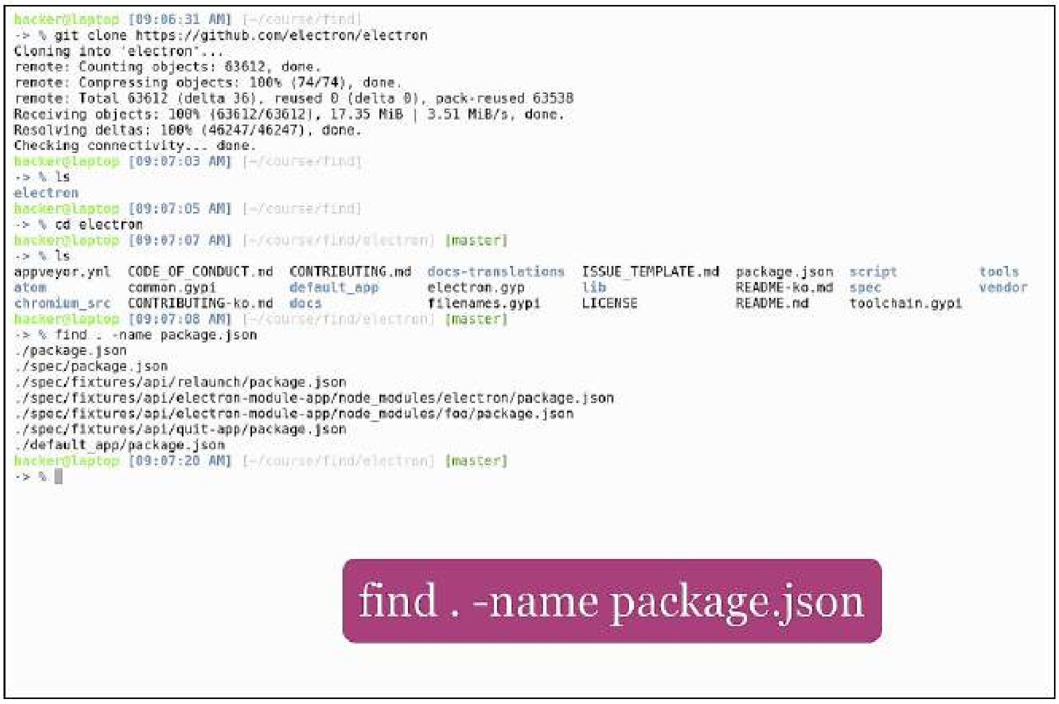
Git clone `https://github.com/electron/electron`

And `cd` into it:

`cd electron`

We see here lots of different files and folders, just like in any normal sized software project. In order to find a particular file, let's say `package.json`, we will use:

`find . -name package.json`



```
hacker@laptop [09:06:31 AM] [~/course/find]
-> % git clone https://github.com/electron/electron
Cloning into 'electron'...
remote: Counting objects: 63612, done.
remote: Compressing objects: 100% (74/74), done.
remote: Total 63612 (delta 36), reused 0 (delta 0), pack-reused 63538
Receiving objects: 100% (63612/63612), 17.35 MiB | 3.51 MiB/s, done.
Resolving deltas: 100% (46247/46247), done.
Checking connectivity... done.
hacker@laptop [09:07:03 AM] [~/course/find]
-> % ls
electron
hacker@laptop [09:07:05 AM] [~/course/find]
-> % cd electron
hacker@laptop [09:07:07 AM] [~/course/find/electron] [master]
-> % ls
appveyor.yml  CODE_OF_CONDUCT.md  CONTRIBUTING.md  docs-translations  ISSUE_TEMPLATE.md  package.json  script          tools
atom          common.gypi         default_app      electron.gyp       lib                README-ko.md  spec            vendor
chromium_src  CONTRIBUTING-ko.md  docs             filenames.gypi     LICENSE            README.md     toolchain.gypi
hacker@laptop [09:07:08 AM] [~/course/find/electron] [master]
-> % find . -name package.json
./package.json
./spec/package.json
./spec/fixtures/api/relaunch/package.json
./spec/fixtures/api/electron-module-app/node_modules/electron/package.json
./spec/fixtures/api/electron-module-app/node_modules/foo/package.json
./spec/fixtures/api/quit-app/package.json
./default_app/package.json
hacker@laptop [09:07:20 AM] [~/course/find/electron] [master]
-> %
```

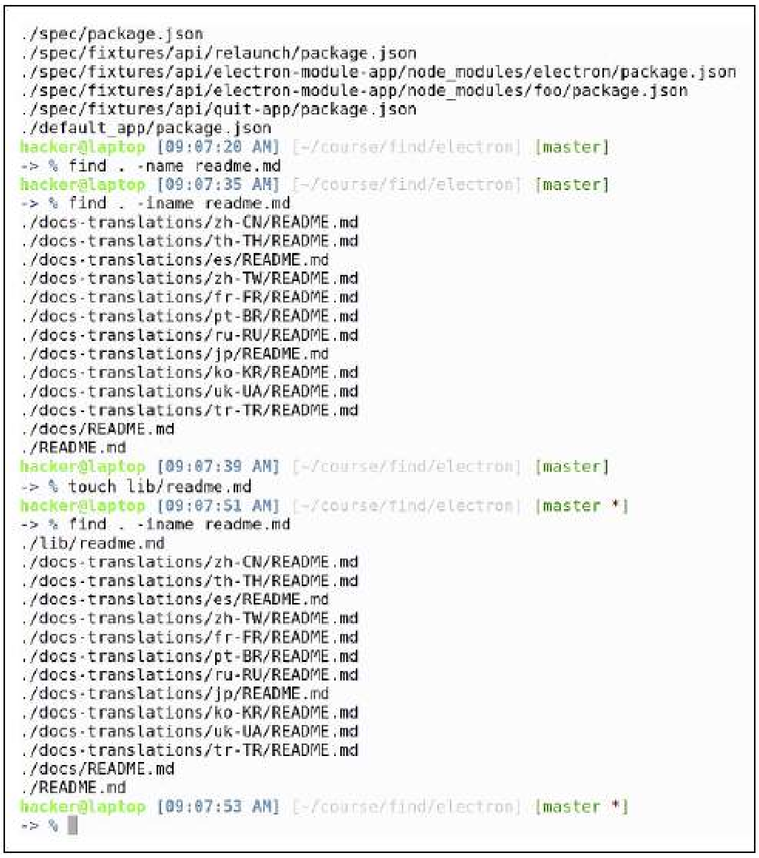
- `.`: This starts the search in the current folder
- `-name`: This helps to search the file name

If we were to look for all readme files in the project, the previous command format is not helpful. We need to issue a case insensitive find. For demonstration purposes, we will also create a `readme.md` file:

`touch lib/readme.md`

We will also use the `-iname` argument for case insensitive search:

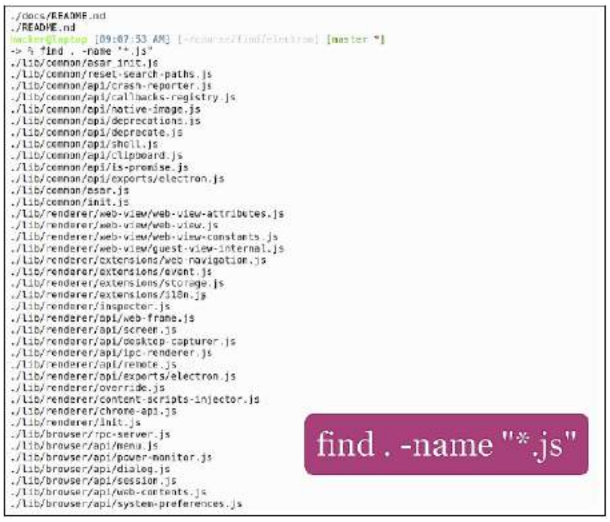
`find . -iname readme.md`



```
./spec/package.json
./spec/fixtures/api/relaunch/package.json
./spec/fixtures/api/electron-module-app/node_modules/electron/package.json
./spec/fixtures/api/electron-module-app/node_modules/foo/package.json
./spec/fixtures/api/quit-app/package.json
./default_app/package.json
hacker@laptop [09:07:20 AM] [~/course/find/electron] [master]
-> % find . -name readme.md
hacker@laptop [09:07:35 AM] [~/course/find/electron] [master]
-> % find . -iname readme.md
./docs-translations/zh-CN/README.md
./docs-translations/th-TH/README.md
./docs-translations/es/README.md
./docs-translations/zh-TW/README.md
./docs-translations/fr-FR/README.md
./docs-translations/pt-BR/README.md
./docs-translations/ru-RU/README.md
./docs-translations/jp/README.md
./docs-translations/ko-KR/README.md
./docs-translations/uk-UA/README.md
./docs-translations/tr-TR/README.md
./docs/README.md
./README.md
hacker@laptop [09:07:39 AM] [~/course/find/electron] [master]
-> % touch lib/readme.md
hacker@laptop [09:07:51 AM] [~/course/find/electron] [master *]
-> % find . -iname readme.md
./lib/readme.md
./docs-translations/zh-CN/README.md
./docs-translations/th-TH/README.md
./docs-translations/es/README.md
./docs-translations/zh-TW/README.md
./docs-translations/fr-FR/README.md
./docs-translations/pt-BR/README.md
./docs-translations/ru-RU/README.md
./docs-translations/jp/README.md
./docs-translations/ko-KR/README.md
./docs-translations/uk-UA/README.md
./docs-translations/tr-TR/README.md
./docs/README.md
./README.md
hacker@laptop [09:07:53 AM] [~/course/find/electron] [master *]
-> %
```

You see here that both `readme.md` and `README.md` have been found. Now, if we were to search for all JavaScript files we would use:

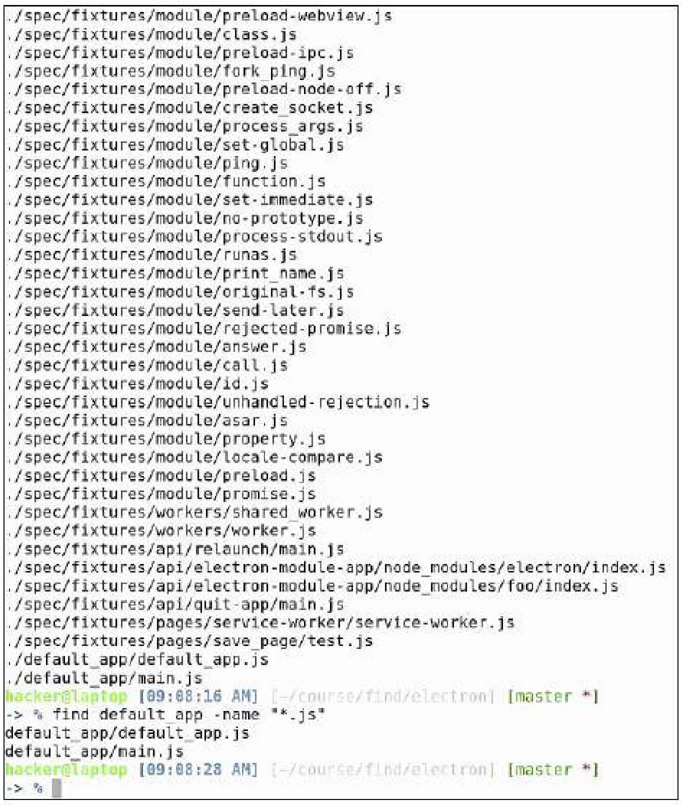
`find . -name "*.js"`



```
./docs/README.md
./README.md
hacker@laptop [09:07:53 AM] [~/course/find/electron] [master *]
-> % find . -name "*.js"
./lib/common/asar_init.js
./lib/common/reset-search-paths.js
./lib/common/api/crash-reporter.js
./lib/common/api/callbacks-registry.js
./lib/common/api/native-image.js
./lib/common/api/deprecations.js
./lib/common/api/deprecate.js
./lib/common/api/shell.js
./lib/common/api/clipboard.js
./lib/common/api/is-promise.js
./lib/common/api/exports/electron.js
./lib/common/asar.js
./lib/common/init.js
./lib/renderer/web-view/web-view-attributes.js
./lib/renderer/web-view/web-view.js
./lib/renderer/web-view/web-view-constants.js
./lib/renderer/web-view/guest-view-internal.js
./lib/renderer/extensions/web-navigation.js
./lib/renderer/extensions/event.js
./lib/renderer/extensions/storage.js
./lib/renderer/extensions/i18n.js
./lib/renderer/inspector.js
./lib/renderer/api/web-frame.js
./lib/renderer/api/screen.js
./lib/renderer/api/desktop-capturer.js
./lib/renderer/api/ipc-renderer.js
./lib/renderer/api/remote.js
./lib/renderer/api/exports/electron.js
./lib/renderer/override.js
./lib/renderer/content-scripts-injector.js
./lib/renderer/chrome-api.js
./lib/renderer/init.js
./lib/browser/rpc-server.js
./lib/browser/api/menu.js
./lib/browser/api/power-monitor.js
./lib/browser/api/dialog.js
./lib/browser/api/session.js
./lib/browser/api/web-contents.js
./lib/browser/api/system-preferences.js
```

And as you can see, there are quite a few results. For narrowing down our results, let's limit the find to the `default_app` folder:

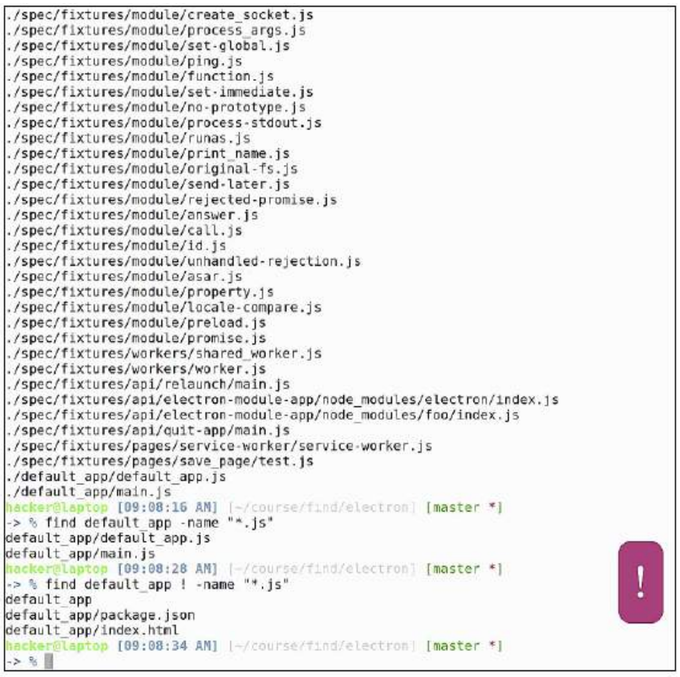
`find default_app -name "*.js"`



```
./spec/fixtures/module/preload-webview.js
./spec/fixtures/module/class.js
./spec/fixtures/module/preload-ipc.js
./spec/fixtures/module/fork_ping.js
./spec/fixtures/module/preload-node-off.js
./spec/fixtures/module/create_socket.js
./spec/fixtures/module/process_args.js
./spec/fixtures/module/set-global.js
./spec/fixtures/module/ping.js
./spec/fixtures/module/function.js
./spec/fixtures/module/set-immediate.js
./spec/fixtures/module/no-prototype.js
./spec/fixtures/module/process-stdout.js
./spec/fixtures/module/runas.js
./spec/fixtures/module/print_name.js
./spec/fixtures/module/original-fs.js
./spec/fixtures/module/send-later.js
./spec/fixtures/module/rejected-promise.js
./spec/fixtures/module/answer.js
./spec/fixtures/module/call.js
./spec/fixtures/module/id.js
./spec/fixtures/module/unhandled-rejection.js
./spec/fixtures/module/asar.js
./spec/fixtures/module/property.js
./spec/fixtures/module/locale-compare.js
./spec/fixtures/module/preload.js
./spec/fixtures/module/promise.js
./spec/fixtures/workers/shared_worker.js
./spec/fixtures/workers/worker.js
./spec/fixtures/api/relaunch/main.js
./spec/fixtures/api/electron-module-app/node_modules/electron/index.js
./spec/fixtures/api/electron-module-app/node_modules/foo/index.js
./spec/fixtures/api/quit-app/main.js
./spec/fixtures/pages/service-worker/service-worker.js
./spec/fixtures/pages/save_page/test.js
./default_app/default_app.js
./default_app/main.js
hacker@laptop [09:08:16 AM] [~/course/find/electron] [master *]
-> % find default_app -name "*.js"
default_app/default_app.js
default_app/main.js
hacker@laptop [09:08:28 AM] [~/course/find/electron] [master *]
-> %
```

As you can see, there are only two `js` files in this folder. And if we were to find all files that are not JavaScript, just add a ! mark before the name argument:

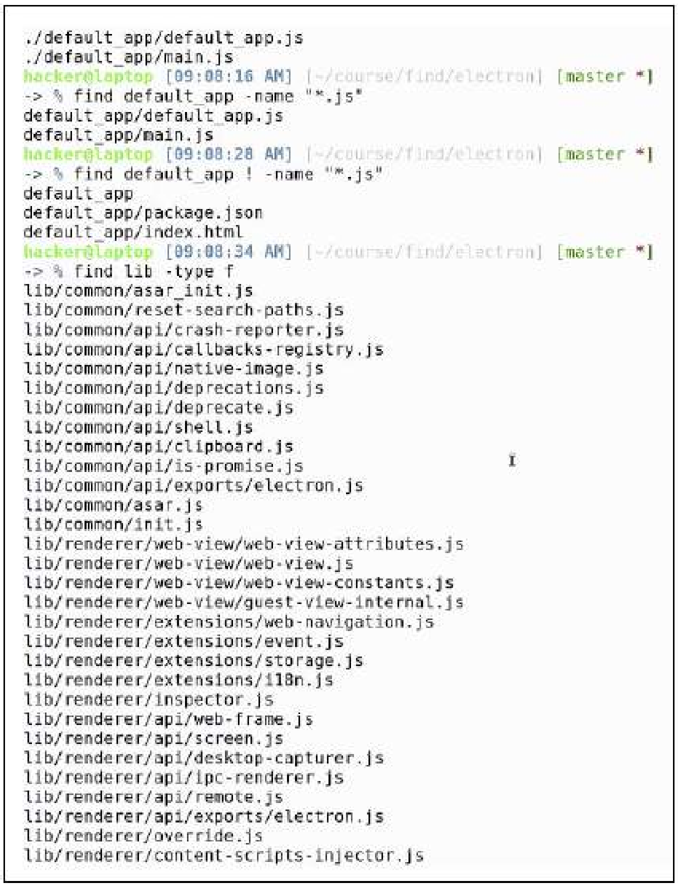
`find default_app ! -name "*.js"`



```
./spec/fixtures/module/create_socket.js
./spec/fixtures/module/process_args.js
./spec/fixtures/module/set-global.js
./spec/fixtures/module/ping.js
./spec/fixtures/module/function.js
./spec/fixtures/module/set-immediate.js
./spec/fixtures/module/no-prototype.js
./spec/fixtures/module/process-stdout.js
./spec/fixtures/module/runas.js
./spec/fixtures/module/print_name.js
./spec/fixtures/module/original-fs.js
./spec/fixtures/module/send-later.js
./spec/fixtures/module/rejected-promise.js
./spec/fixtures/module/answer.js
./spec/fixtures/module/call.js
./spec/fixtures/module/id.js
./spec/fixtures/module/unhandled-rejection.js
./spec/fixtures/module/asar.js
./spec/fixtures/module/property.js
./spec/fixtures/module/locale-compare.js
./spec/fixtures/module/preload.js
./spec/fixtures/module/promise.js
./spec/fixtures/workers/shared_worker.js
./spec/fixtures/workers/worker.js
./spec/fixtures/api/relaunch/main.js
./spec/fixtures/api/electron-module-app/node_modules/electron/index.js
./spec/fixtures/api/electron-module-app/node_modules/foo/index.js
./spec/fixtures/api/quit-app/main.js
./spec/fixtures/pages/service-worker/service-worker.js
./spec/fixtures/pages/save_page/test.js
./default_app/default_app.js
./default_app/main.js
hacker@laptop [09:08:16 AM] [~/course/find/electron] [master *]
-> % find default_app -name "*.js"
default_app/default_app.js
default_app/main.js
hacker@laptop [09:08:28 AM] [~/course/find/electron] [master *]
-> % find default_app ! -name "*.js"
default_app
default_app/package.json
default_app/index.html
hacker@laptop [09:08:34 AM] [~/course/find/electron] [master *]
-> %
```

You can see here all files that don't end their name with js. If we were to look for all inodes in the directory, which are of type file, we would use the `-type f` argument:

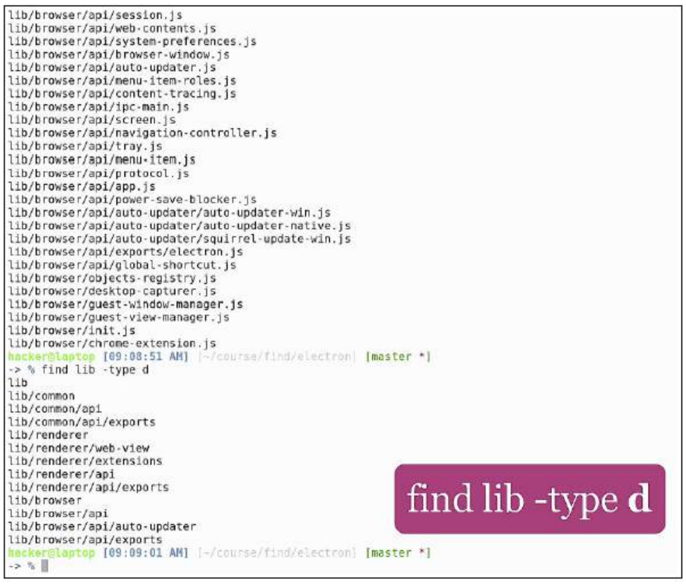
`find lib -type f`



```
./default_app/default_app.js
./default_app/main.js
hacker@laptop [09:08:16 AM] [~/course/find/electron] [master *]
-> % find default_app -name "*.js"
default_app/default_app.js
default_app/main.js
hacker@laptop [09:08:28 AM] [~/course/find/electron] [master *]
-> % find default_app ! -name "*.js"
default_app
default_app/package.json
default_app/index.html
hacker@laptop [09:08:34 AM] [~/course/find/electron] [master *]
-> % find lib -type f
lib/common/asar_init.js
lib/common/reset-search-paths.js
lib/common/api/crash-reporter.js
lib/common/api/callbacks-registry.js
lib/common/api/native-image.js
lib/common/api/deprecations.js
lib/common/api/deprecate.js
lib/common/api/shell.js
lib/common/api/clipboard.js
lib/common/api/is-promise.js
lib/common/api/exports/electron.js
lib/common/asar.js
lib/common/init.js
lib/renderer/web-view/web-view-attributes.js
lib/renderer/web-view/web-view.js
lib/renderer/web-view/web-view-constants.js
lib/renderer/web-view/guest-view-internal.js
lib/renderer/extensions/web-navigation.js
lib/renderer/extensions/event.js
lib/renderer/extensions/storage.js
lib/renderer/extensions/i18n.js
lib/renderer/inspector.js
lib/renderer/api/web-frame.js
lib/renderer/api/screen.js
lib/renderer/api/desktop-capturer.js
lib/renderer/api/ipc-renderer.js
lib/renderer/api/remote.js
lib/renderer/api/exports/electron.js
lib/renderer/override.js
lib/renderer/content-scripts-injector.js
```

In the same way, we'd use `-type d` to find all directories in a specific location:

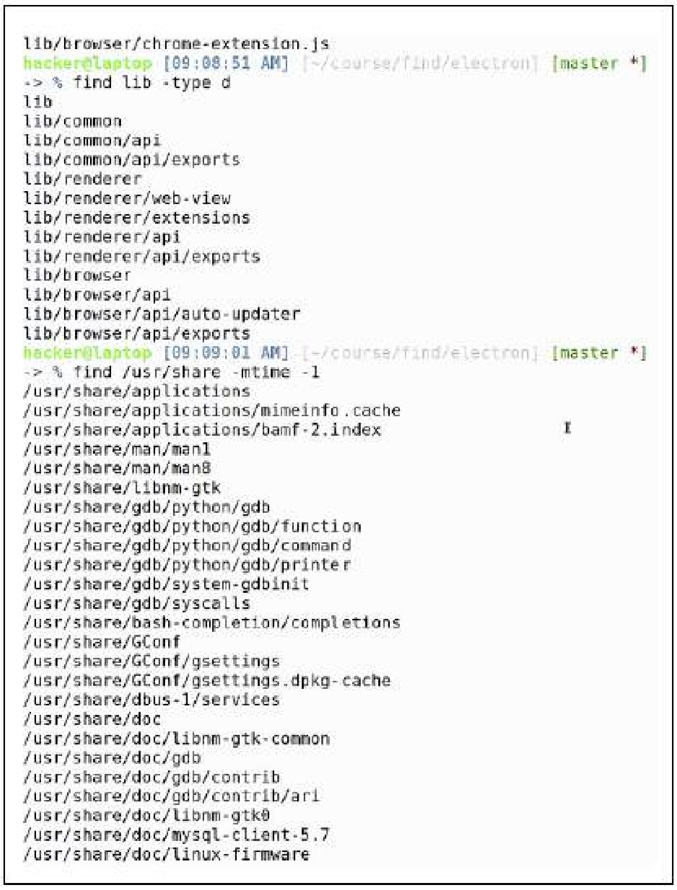
`find lib -type d`



```
lib/browser/api/session.js
lib/browser/api/web-contents.js
lib/browser/api/system-preferences.js
lib/browser/api/browser-window.js
lib/browser/api/auto-updater.js
lib/browser/api/menu-item-roles.js
lib/browser/api/content-tracing.js
lib/browser/api/ipc-main.js
lib/browser/api/screen.js
lib/browser/api/navigation-controller.js
lib/browser/api/tray.js
lib/browser/api/menu-item.js
lib/browser/api/protocol.js
lib/browser/api/app.js
lib/browser/api/power-save-blocker.js
lib/browser/api/auto-updater/auto-updater-win.js
lib/browser/api/auto-updater/auto-updater-native.js
lib/browser/api/auto-updater/squirrel-update-win.js
lib/browser/api/exports/electron.js
lib/browser/api/global-shortcut.js
lib/browser/objects-registry.js
lib/browser/desktop-capturer.js
lib/browser/guest-window-manager.js
lib/browser/guest-view-manager.js
lib/browser/init.js
lib/browser/chrome-extension.js
hacker@laptop [09:08:51 AM] [~/course/find/electron] [master *]
-> % find lib -type d
lib
lib/common
lib/common/api
lib/common/api/exports
lib/renderer
lib/renderer/web-view
lib/renderer/extensions
lib/renderer/api
lib/renderer/api/exports
lib/browser
lib/browser/api
lib/browser/api/auto-updater
lib/browser/api/exports
hacker@laptop [09:09:01 AM] [~/course/find/electron] [master *]
-> %
```

Find can also locate files based on time identifiers. For example, in order to find all files in the `/usr/share` directory that were modified in the last 24 hours, issue the following command:

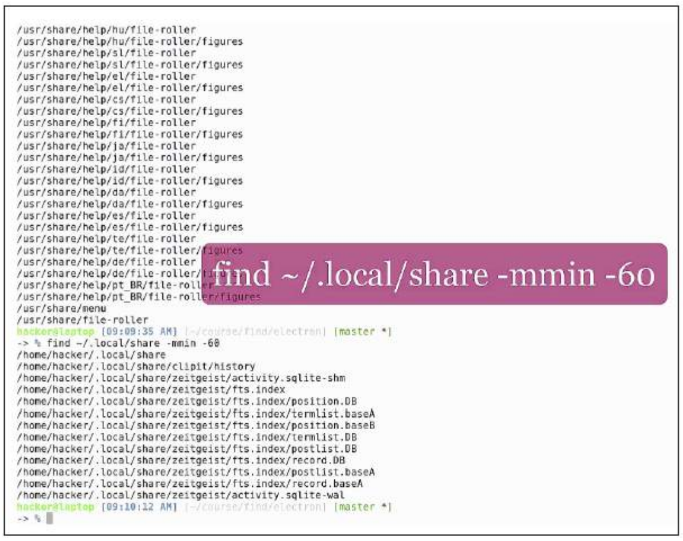
`find /usr/share -mtime -1`



```
lib/browser/chrome-extension.js
hacker@laptop [09:08:51 AM] [~/course/find/electron] [master *]
-> % find lib -type d
lib
lib/common
lib/common/api
lib/common/api/exports
lib/renderer
lib/renderer/web-view
lib/renderer/extensions
lib/renderer/api
lib/renderer/api/exports
lib/browser
lib/browser/api
lib/browser/api/auto-updater
lib/browser/api/exports
hacker@laptop [09:09:01 AM] [~/course/find/electron] [master *]
-> % find /usr/share -mtime -1
/usr/share/applications
/usr/share/applications/mimeinfo.cache
/usr/share/applications/bamf-2.index
/usr/share/man/man1
/usr/share/man/man8
/usr/share/libnm-gtk
/usr/share/gdb/python/gdb
/usr/share/gdb/python/gdb/function
/usr/share/gdb/python/gdb/command
/usr/share/gdb/python/gdb/printer
/usr/share/gdb/system-gdbinit
/usr/share/gdb/syscalls
/usr/share/bash-completion/completions
/usr/share/GConf
/usr/share/GConf/gsettings
/usr/share/GConf/gsettings.dpkg-cache
/usr/share/dbus-1/services
/usr/share/doc
/usr/share/doc/libnm-gtk-common
/usr/share/doc/gdb
/usr/share/doc/gdb/contrib
/usr/share/doc/gdb/contrib/ari
/usr/share/doc/libnm-gtk0
/usr/share/doc/mysql-client-5.7
/usr/share/doc/linux-firmware
```

I have quite a big list. You can see the `-mtime -3` broadens the list even more. If we were to find, for example, all the files modified in the last hour, we can use
`-mmin -60`:

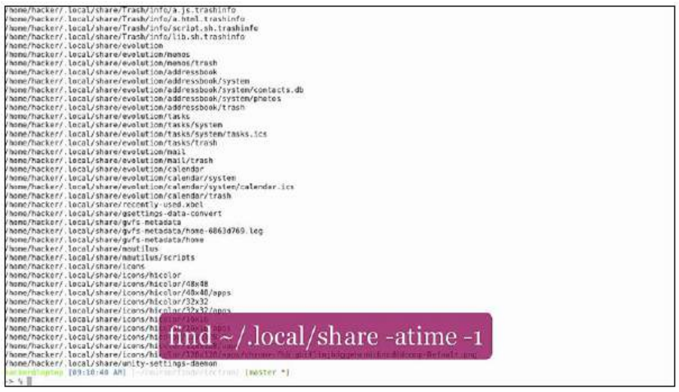
`find ~/.local/share -mmin -60`



```
/usr/share/help/hu/file-roller
/usr/share/help/hu/file-roller/figures
/usr/share/help/sl/file-roller
/usr/share/help/sl/file-roller/figures
/usr/share/help/el/file-roller
/usr/share/help/el/file-roller/figures
/usr/share/help/cs/file-roller
/usr/share/help/cs/file-roller/figures
/usr/share/help/fi/file-roller
/usr/share/help/fi/file-roller/figures
/usr/share/help/ja/file-roller
/usr/share/help/ja/file-roller/figures
/usr/share/help/id/file-roller
/usr/share/help/id/file-roller/figures
/usr/share/help/da/file-roller
/usr/share/help/da/file-roller/figures
/usr/share/help/es/file-roller
/usr/share/help/es/file-roller/figures
/usr/share/help/te/file-roller
/usr/share/help/te/file-roller/figures
/usr/share/help/de/file-roller
/usr/share/help/de/file-roller/figures
/usr/share/help/pt_BR/file-roller
/usr/share/help/pt_BR/file-roller/figures
/usr/share/menu
/usr/share/file-roller
hacker@laptop [09:09:35 AM] [~/course/find/electron] [master *]
-> % find ~/.local/share -mmin -60
/home/hacker/.local/share
/home/hacker/.local/share/clipit/history
/home/hacker/.local/share/zeitgeist/activity.sqlite-shm
/home/hacker/.local/share/zeitgeist/fts.index
/home/hacker/.local/share/zeitgeist/fts.index/position.DB
/home/hacker/.local/share/zeitgeist/fts.index/termlist.baseA
/home/hacker/.local/share/zeitgeist/fts.index/position.baseB
/home/hacker/.local/share/zeitgeist/fts.index/termlist.DB
/home/hacker/.local/share/zeitgeist/fts.index/postlist.DB
/home/hacker/.local/share/zeitgeist/fts.index/record.DB
/home/hacker/.local/share/zeitgeist/fts.index/postlist.baseA
/home/hacker/.local/share/zeitgeist/fts.index/record.baseA
/home/hacker/.local/share/zeitgeist/activity.sqlite-wal
hacker@laptop [09:10:12 AM] [~/course/find/electron] [master *]
-> %
```

A good folder to search is `~/.local/share`, If we use `-mmin -90`, the list broadens again. Find can also show us the list of files accessed in the last 24 hours by using the `-atime -1` argument like so:

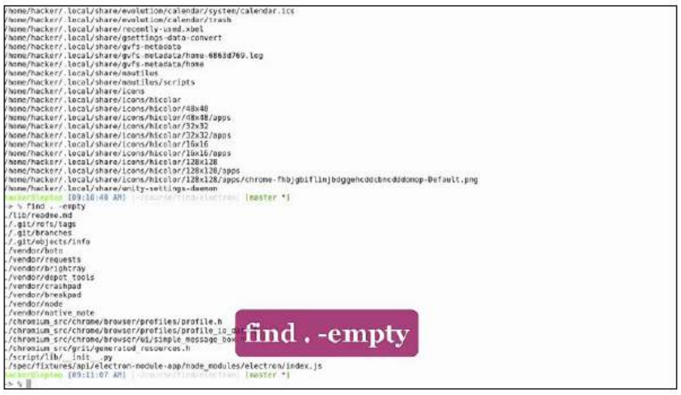
`find ~/.local/share -atime -1`



```
/home/hacker/.local/share/Trash/info/a.js.trashinfo
/home/hacker/.local/share/Trash/info/a.html.trashinfo
/home/hacker/.local/share/Trash/info/script.sh.trashinfo
/home/hacker/.local/share/Trash/info/lib.sh.trashinfo
/home/hacker/.local/share/evolution
/home/hacker/.local/share/evolution/memos
/home/hacker/.local/share/evolution/memos/trash
/home/hacker/.local/share/evolution/addressbook
/home/hacker/.local/share/evolution/addressbook/system
/home/hacker/.local/share/evolution/addressbook/system/contacts.db
/home/hacker/.local/share/evolution/addressbook/system/photos
/home/hacker/.local/share/evolution/addressbook/trash
/home/hacker/.local/share/evolution/tasks
/home/hacker/.local/share/evolution/tasks/system
/home/hacker/.local/share/evolution/tasks/system/tasks.ics
/home/hacker/.local/share/evolution/tasks/trash
/home/hacker/.local/share/evolution/mail
/home/hacker/.local/share/evolution/mail/trash
/home/hacker/.local/share/evolution/calendar
/home/hacker/.local/share/evolution/calendar/system
/home/hacker/.local/share/evolution/calendar/system/calendar.ics
/home/hacker/.local/share/evolution/calendar/trash
/home/hacker/.local/share/recently-used.xbel
/home/hacker/.local/share/gsettings-data-convert
/home/hacker/.local/share/gvfs-metadata
/home/hacker/.local/share/gvfs-metadata/home-6863d769.log
/home/hacker/.local/share/gvfs-metadata/home
/home/hacker/.local/share/nautilus
/home/hacker/.local/share/nautilus/scripts
/home/hacker/.local/share/icons
/home/hacker/.local/share/icons/hicolor
/home/hacker/.local/share/icons/hicolor/48x48
/home/hacker/.local/share/icons/hicolor/48x48/apps
/home/hacker/.local/share/icons/hicolor/32x32
/home/hacker/.local/share/icons/hicolor/32x32/apps
/home/hacker/.local/share/icons/hi[illegible]
/home/hacker/.local/share/icons/hi[illegible]
/home/hacker/.local/share/icons/hi[illegible]
/home/hacker/.local/share/icons/hi[illegible]
/home/hacker/.local/share/icons/hi[illegible]
/home/hacker/.local/share/unity-settings-daemon
hacker@laptop [09:10:48 AM] [~/course/find/electron] [master *]
-> %
```

While working with lots of project files, if sometimes the case in some projects remain empty, and we forget to delete them. In order to locate all empty files just do a:

`find . -empty`



```
/home/hacker/.local/share/evolution/calendar/system/calendar.ics
/home/hacker/.local/share/evolution/calendar/trash
/home/hacker/.local/share/recently-used.xbel
/home/hacker/.local/share/gsettings-data-convert
/home/hacker/.local/share/gvfs-metadata
/home/hacker/.local/share/gvfs-metadata/home-6863d769.log
/home/hacker/.local/share/gvfs-metadata/home
/home/hacker/.local/share/nautilus
/home/hacker/.local/share/nautilus/scripts
/home/hacker/.local/share/icons
/home/hacker/.local/share/icons/hicolor
/home/hacker/.local/share/icons/hicolor/48x48
/home/hacker/.local/share/icons/hicolor/48x48/apps
/home/hacker/.local/share/icons/hicolor/32x32
/home/hacker/.local/share/icons/hicolor/32x32/apps
/home/hacker/.local/share/icons/hicolor/16x16
/home/hacker/.local/share/icons/hicolor/16x16/apps
/home/hacker/.local/share/icons/hicolor/128x128
/home/hacker/.local/share/icons/hicolor/128x128/apps
/home/hacker/.local/share/icons/hicolor/128x128/apps/chrome-fhbjgbiflinjbdggehcddcbncdddomop-Default.png
/home/hacker/.local/share/unity-settings-daemon
hacker@laptop [09:10:48 AM] [~/course/find/electron] [master *]
-> % find . -empty
./lib/readme.md
./.git/refs/tags
./.git/branches
./.git/objects/info
./vendor/boto
./vendor/requests
./vendor/brightray
./vendor/depot_tools
./vendor/crashpad
./vendor/breakpad
./vendor/node
./vendor/native_mate
./chromium_src/chrome/browser/profiles/profile.h
./chromium_src/chrome/browser/profiles/profile_io_[illegible]
./chromium_src/chrome/browser/ui/simple_message_box[illegible]
./chromium_src/grit/generated_resources.h
./script/lib/__init__.py
./spec/fixtures/api/electron-module-app/node_modules/electron/index.js
hacker@laptop [09:11:07 AM] [~/course/find/electron] [master *]
-> %
```

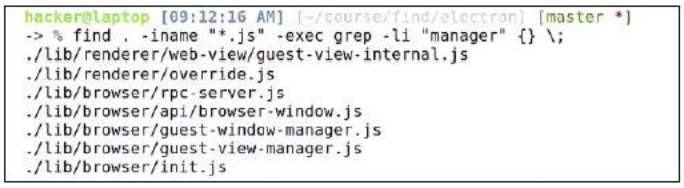
Removing empty files will keep our project clean, but when it comes to reducing size, we sometimes want to know which files are taking up most of the space. Find can also do searches based on file size. For example, let's find all the files larger than 1 mega:

`find . -size +1M`

use -1M for smaller.

if we want to find all `javascript` files that contain the text `manager`, we can combine find with grep, command as follows:

`find . -name "*.js" -exec grep -li 'manager' {} \;`



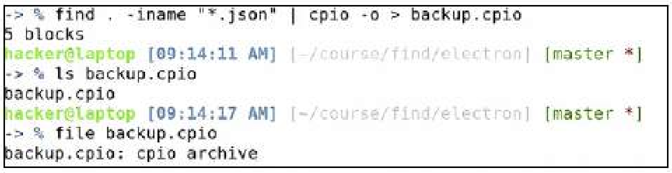
```
hacker@laptop [09:12:16 AM] [~/course/find/electron] [master *]
-> % find . -iname "*.js" -exec grep -li "manager" {} \;
./lib/renderer/web-view/guest-view-internal.js
./lib/renderer/override.js
./lib/browser/rpc-server.js
./lib/browser/api/browser-window.js
./lib/browser/guest-window-manager.js
./lib/browser/guest-view-manager.js
./lib/browser/init.js
```

Moving on with the practical examples, let's say you have a folder where you want to remove all the files modified in the last 100 days. We can see our `default_app` folder contains such files. If we combine find with `rm` like so:

`find default_app -mtime -100 -exec rm -rf {} \;`

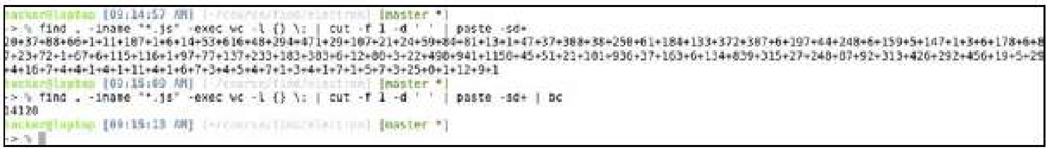
We can do a quick cleanup. Find can be used for smart backups. For example, if we were to backup all `json` files in the project we would combine find with the `cpio` backup utility using a pipe and a standard output redirection:

`find . -name "*.json" | cpio -o > backup.cpio`



```
-> % find . -iname "*.json" | cpio -o > backup.cpio
5 blocks
hacker@laptop [09:14:11 AM] [~/course/find/electron] [master *]
-> % ls backup.cpio
backup.cpio
hacker@laptop [09:14:17 AM] [~/course/find/electron] [master *]
-> % file backup.cpio
backup.cpio: cpio archive
```

- In order to do this, we combine find with `wc -l`:
  `find . -iname "*.js" -exec wc -l {} \;`
- This will give us all js files and the number of lines. We can pipe this to cut:
  `find . -iname "*.js" -exec wc -l {} \; | cut -f 1 -d ' '`
- To only output the number of lines, and then pipe to the paste command, we do this:
  `find . -iname "*.js" -exec wc -l {} \; | cut -f 1 -d ' ' |`
`paste -sd+`
- The above will merge all our lines with the + sign as a delimiter. This, of course, can translate to an arithmetic operation, which we can calculate using the binary calculator (`bc`):
  `find . -iname "*.js" -exec wc -l {} \; | cut -f 1 -d ' ' |`
`paste -sd+ | bc`

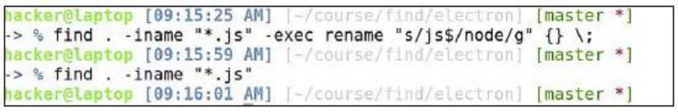


```
hacker@laptop [09:14:57 AM] [~/course/find/electron] [master *]
-> % find . -iname "*.js" -exec wc -l {} \; | cut -f 1 -d ' ' | paste -sd+
20+37+88+66+1+11+187+1+6+14+53+616+48+294+471+29+107+21+24+59+84+81+13+1+47+37+388+38+258+61+184+133+372+387+6+197+44+248+6+159+5+147+1+3+6+178+6+8
7+23+72+1+67+6+115+116+1+97+77+137+233+183+383+6+12+80+3+22+498+941+1150+45+51+23+101+936+37+163+6+134+839+315+27+248+87+92+313+426+292+456+19+5+29
+4+16+7+4+4+1+4+1+11+4+1+6+7+3+4+5+4+7+1+3+4+1+7+1+5+7+3+25+0+1+12+9+1
hacker@laptop [09:15:09 AM] [~/course/find/electron] [master *]
-> % find . -iname "*.js" -exec wc -l {} \; | cut -f 1 -d ' ' | paste -sd+ | bc
14120
hacker@laptop [09:15:13 AM] [~/course/find/electron] [master *]
-> %
```

In order to mass rename files, like changing the file extension name to `node` for all `js` files we can use this command:

`find . -type f -iname "*.js" -exec rename "s/js$/node/g" {} \;`

You can see the rename syntax is quite similar to sed. In addition, there are no more `.js` files left, as all have been renamed to `.node`:



```
hacker@laptop [09:15:25 AM] [~/course/find/electron] [master *]
-> % find . -iname "*.js" -exec rename "s/js$/node/g" {} \;
hacker@laptop [09:15:59 AM] [~/course/find/electron] [master *]
-> % find . -iname "*.js"
hacker@laptop [09:16:01 AM] [~/course/find/electron] [master *]
```

<u>Tmux</u>

- To get started with `tmux` on Ubuntu, we first need to install it:
- **`sudo apt install tmux`**
- **`tmux`** : command to run the tool
- **`tmux`** `ls`: to list the session
- **`tmux new -s mysession`**: to start the session
- To list and switch `tmux` sessions inside `tmux`, hit *Ctrl* + *B S*.
- leave a session running and go back to the normal terminal hit *Ctrl* + *b d*
- **`tmux kill-session -t mysession`**: to kill the session
- *Ctrl* + *b* + " : split the screen horizontally

- *Ctrl + b + %* : split the screen vertically
- *Ctrl + b c*: to create window
- *Ctrl + b w*: make a list
- *Ctrl + b &*: delete

Network

- **`netstat -plnt`** : show all open ports on our host
- **`python -m SimpleHTTPServer`**: Let's open the most basic HTTP server
- **`sudo vim /etc/hosts`** : to connect to servers and the servers change their IP address, or when you want to emulate a remote server on a local machine.
- **`sudo apt install nmap`**: to install the tool

Autoenv

- To install it we go to the official GitHub page and follow the instructions: https://github.com/kennethreitz/autoenv
- To load the tool (autoenv): **`source ~/.autoenv/activate.sh`**
- To open an environment file: **`vim project1/.env`**
- To call and set variable named ENV : **`export ENV=dev`**
- Hit y to load the file.
-

Laboratory Procotol Developer and Supervisor(s) Information
Protocol Developer : Arli Aditya Parikesit
Email : arli.parikesit@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   q. If you have any troubleshoot please contact your supervisor or Building Management
   r. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   s. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   t. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 5

Date : October 1st, 2019

Laboratory :Bioinformatics laboratory

Overview

ImageMagick is used to process all our images automatically. Git flow branching model is utilized to help merging git conflicts using command line. It also can save the day by proxying requests coming from the internet to our laptop. We will also explore the versatile query capabilities with JSON.

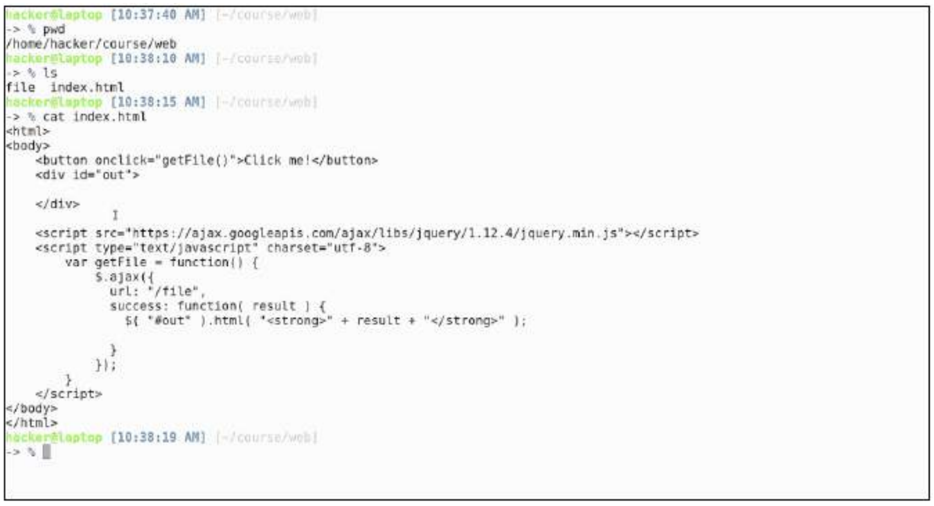
Objective

Students are expected to understand on how applying Linux development tools.

Procedure

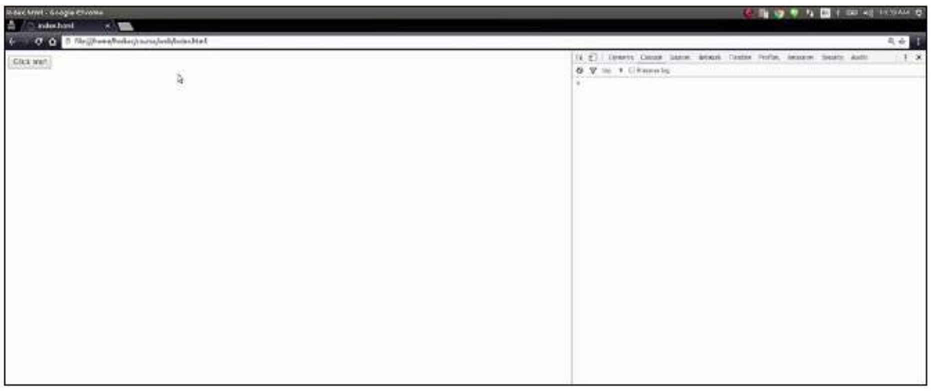
Building a web server

- We have prepared a basic demo `html` file that contains a button, a `div`, a `jquery` function (for helping us do some `ajax` calls) and a script that will try to load static content from our server and put the content inside the `div` tag
- The script is trying to load a simple text file on the disk, `/file`:

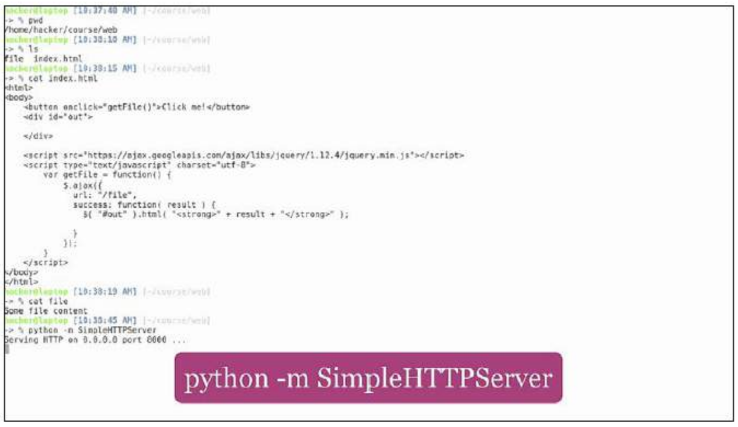


```
hacker@laptop [10:37:40 AM] [~/course/web]
-> % pwd
/home/hacker/course/web
hacker@laptop [10:38:10 AM] [~/course/web]
-> % ls
file  index.html
hacker@laptop [10:38:15 AM] [~/course/web]
-> % cat index.html
<html>
<body>
    <button onclick="getFile()">Click me!</button>
    <div id="out">

    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script type="text/javascript" charset="utf-8">
        var getFile = function() {
            $.ajax({
              url: "/file",
              success: function( result ) {
                $( "#out" ).html( "<strong>" + result + "</strong>" );

              }
            });
        }
    </script>
</body>
</html>
hacker@laptop [10:38:19 AM] [~/course/web]
-> % 
```

- If we open this file inside our browser, we can see the page content:



- In order to start an HTTP server in the same folder as the file, we type the following command:
  `python -m SimpleHTTPServer`



```
hacker@laptop [10:37:40 AM] [~/course/web]
-> % pwd
/home/hacker/course/web
hacker@laptop [10:38:10 AM] [~/course/web]
-> % ls
file  index.html
hacker@laptop [10:38:15 AM] [~/course/web]
-> % cat index.html
<html>
<body>
    <button onclick="getFile()">Click me!</button>
    <div id="out">

    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <script type="text/javascript" charset="utf-8">
        var getFile = function() {
            $.ajax({
              url: "/file",
              success: function( result ) {
                $( "#out" ).html( "<strong>" + result + "</strong>" );

              }
            });
        }
    </script>
</body>
</html>
hacker@laptop [10:38:19 AM] [~/course/web]
-> % cat file
Some file content
hacker@laptop [10:38:45 AM] [~/course/web]
-> % python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

- To open the address in the browser: Click on the **Click me!** Button

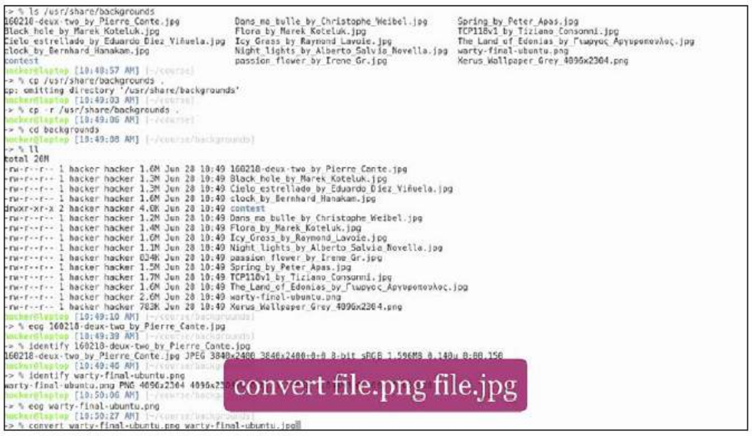
Shrinking spells and other ImageMagick

- To install it, run the following:
  `sudo apt install imagemagick`

```
hacker@laptop [10:48:25 AM] [~/course]
-> % sudo apt install imagemagick
```

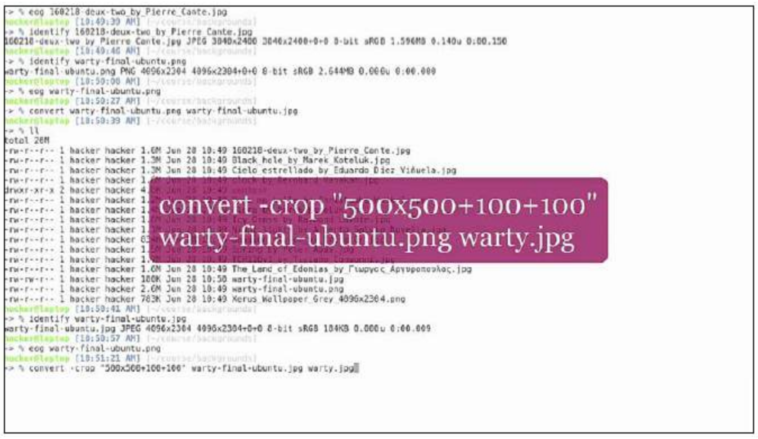
- Let's use the default Ubuntu backgrounds that can be found in `/usr/share/backgrounds`.
- Let's copy the backgrounds to another location so that we don't alter our default ones
- we can see images from `ls` that it is a JPEG image.

- To open it and see how it looks, let's use the **eog** (**eye of gnome**) image viewer
- To process an image and what that image actually is can be found out using ImageMagick’s tool called Identify
  `identify image_name`
- The `warty-final-ubuntu.png` has similar output format with higher size and resolution with PNG format.
  `eog warty-final-ubuntu.png`
- In order to convert from one type to the other, we use the `imagemagick convert` command with two parameters: input filename and output filename:
  `convert file.png file.jpg`



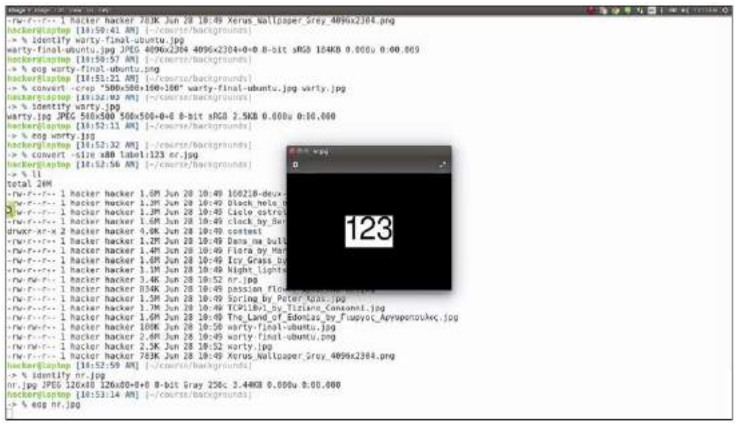
```
-> % ls /usr/share/backgrounds
160218-deux-two_by_Pierre_Cante.jpg          Dans_ma_bulle_by_Christophe_Weibel.jpg      Spring_by_Peter_Apas.jpg
Black_hole_by_Marek_Koteluk.jpg              Flora_by_Marek_Koteluk.jpg                  TCP118v1_by_Tiziano_Consonni.jpg
Cielo_estrellado_by_Eduardo_Diez_Viñuela.jpg Icy_Grass_by_Raymond_Lavoie.jpg             The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
clock_by_Bernhard_Hanakam.jpg                Night_lights_by_Alberto_Salvia_Novella.jpg  warty-final-ubuntu.png
contest                                      passion_flower_by_Irene_Gr.jpg              Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:48:57 AM] [~/course]
-> % cp /usr/share/backgrounds .
cp: omitting directory '/usr/share/backgrounds'
hacker@laptop [10:49:03 AM] [~/course]
-> % cp -r /usr/share/backgrounds .
hacker@laptop [10:49:06 AM] [~/course]
-> % cd backgrounds
hacker@laptop [10:49:08 AM] [~/course/backgrounds]
-> % ll
total 26M
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 160218-deux-two_by_Pierre_Cante.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Black_hole_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Cielo_estrellado_by_Eduardo_Diez_Viñuela.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 clock_by_Bernhard_Hanakam.jpg
drwxr-xr-x 2 hacker hacker 4.0K Jun 28 10:49 contest
-rw-r--r-- 1 hacker hacker 1.2M Jun 28 10:49 Dans_ma_bulle_by_Christophe_Weibel.jpg
-rw-r--r-- 1 hacker hacker 1.4M Jun 28 10:49 Flora_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 Icy_Grass_by_Raymond_Lavoie.jpg
-rw-r--r-- 1 hacker hacker 1.1M Jun 28 10:49 Night_lights_by_Alberto_Salvia_Novella.jpg
-rw-r--r-- 1 hacker hacker 834K Jun 28 10:49 passion_flower_by_Irene_Gr.jpg
-rw-r--r-- 1 hacker hacker 1.5M Jun 28 10:49 Spring_by_Peter_Apas.jpg
-rw-r--r-- 1 hacker hacker 1.7M Jun 28 10:49 TCP118v1_by_Tiziano_Consonni.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
-rw-r--r-- 1 hacker hacker 2.6M Jun 28 10:49 warty-final-ubuntu.png
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:49:10 AM] [~/course/backgrounds]
-> % eog 160218-deux-two_by_Pierre_Cante.jpg
hacker@laptop [10:49:39 AM] [~/course/backgrounds]
-> % identify 160218-deux-two_by_Pierre_Cante.jpg
160218-deux-two_by_Pierre_Cante.jpg JPEG 3840x2400 3840x2400+0+0 8-bit sRGB 1.596MB 0.140u 0:00.150
hacker@laptop [10:49:46 AM] [~/course/backg
-> % identify warty-final-ubuntu.png
warty-final-ubuntu.png PNG 4096x2304 4096x23
hacker@laptop [10:50:06 AM] [~/course/backg
-> % eog warty-final-ubuntu.png
hacker@laptop [10:50:27 AM] [~/course/backgrounds]
-> % convert warty-final-ubuntu.png warty-final-ubuntu.jpg
```

- If we want to crop a region of the image, we can do that with `convert`.
  `convert -crop "500x500+100+100" warty-final-ubuntu.png warty.jpg`



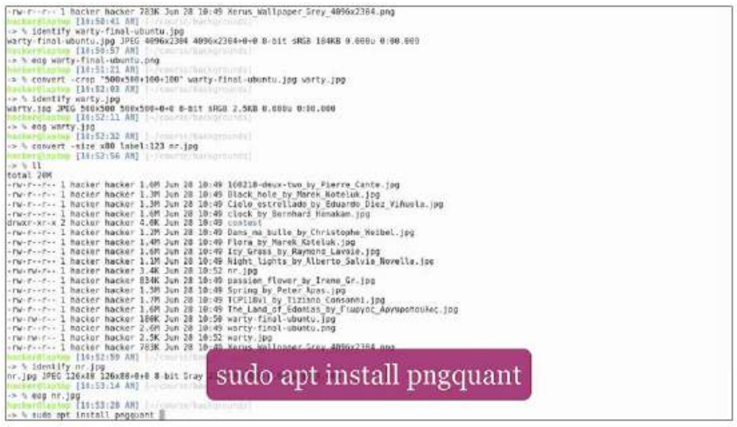
```
-> % eog 160218-deux-two_by_Pierre_Cante.jpg
hacker@laptop [10:49:39 AM] [~/course/backgrounds]
-> % identify 160218-deux-two_by_Pierre_Cante.jpg
160218-deux-two_by_Pierre_Cante.jpg JPEG 3840x2400 3840x2400+0+0 8-bit sRGB 1.596MB 0.140u 0:00.150
hacker@laptop [10:49:46 AM] [~/course/backgrounds]
-> % identify warty-final-ubuntu.png
warty-final-ubuntu.png PNG 4096x2304 4096x2304+0+0 8-bit sRGB 2.644MB 0.000u 0:00.000
hacker@laptop [10:50:00 AM] [~/course/backgrounds]
-> % eog warty-final-ubuntu.png
hacker@laptop [10:50:27 AM] [~/course/backgrounds]
-> % convert warty-final-ubuntu.png warty-final-ubuntu.jpg
hacker@laptop [10:50:39 AM] [~/course/backgrounds]
-> % ll
total 26M
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 160218-deux-two_by_Pierre_Cante.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Black_hole_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Cielo_estrellado_by_Eduardo_Diez_Viñuela.jpg
-rw-r--r-- 1 hacker hacker 1.
drwxr-xr-x 2 hacker hacker 4.
-rw-r--r-- 1 hacker hacker 1.
-rw-r--r-- 1 hacker hacker 1.
-rw-r--r-- 1 hacker hacker 1.
-rw-r--r-- 1 hacker hacker 1.
-rw-r--r-- 1 hacker hacker 83
-rw-r--r-- 1 hacker hacker 1.
-rw-r--r-- 1 hacker hacker 1.
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
-rw-rw-r-- 1 hacker hacker 180K Jun 28 10:50 warty-final-ubuntu.jpg
-rw-r--r-- 1 hacker hacker 2.6M Jun 28 10:49 warty-final-ubuntu.png
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:50:41 AM] [~/course/backgrounds]
-> % identify warty-final-ubuntu.jpg
warty-final-ubuntu.jpg JPEG 4096x2304 4096x2304+0+0 8-bit sRGB 184KB 0.000u 0:00.009
hacker@laptop [10:50:57 AM] [~/course/backgrounds]
-> % eog warty-final-ubuntu.png
hacker@laptop [10:51:21 AM] [~/course/backgrounds]
-> % convert -crop "500x500+100+100" warty-final-ubuntu.jpg warty.jpg
```

- The `convert` command is also good at creating images. If we want to create an image from a text string, we could use the following:
  `convert -size x80 label:123 nr.jpg`



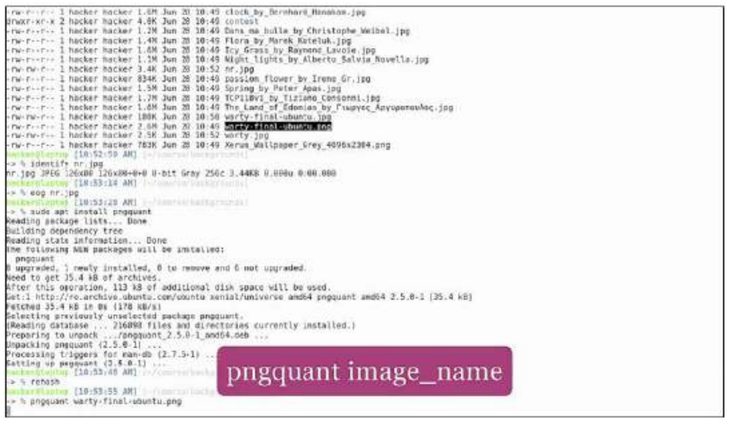
```
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:50:41 AM] [~/course/backgrounds]
-> % identify warty-final-ubuntu.jpg
warty-final-ubuntu.jpg JPEG 4096x2304 4096x2304+0+0 8-bit sRGB 184KB 0.000u 0:00.009
hacker@laptop [10:50:57 AM] [~/course/backgrounds]
-> % eog warty-final-ubuntu.png
hacker@laptop [10:51:21 AM] [~/course/backgrounds]
-> % convert -crop "500x500+100+100" warty-final-ubuntu.jpg warty.jpg
hacker@laptop [10:52:03 AM] [~/course/backgrounds]
-> % identify warty.jpg
warty.jpg JPEG 500x500 500x500+0+0 8-bit sRGB 2.5KB 0.000u 0:00.000
hacker@laptop [10:52:11 AM] [~/course/backgrounds]
-> % eog warty.jpg
hacker@laptop [10:52:32 AM] [~/course/backgrounds]
-> % convert -size x80 label:123 nr.jpg
hacker@laptop [10:52:56 AM] [~/course/backgrounds]
-> % ll
total 20M
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 160218-deux-
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Black_hole_b
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Cielo_estrel
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 clock_by_Ber
drwxr-xr-x 2 hacker hacker 4.0K Jun 28 10:49 contest
-rw-r--r-- 1 hacker hacker 1.2M Jun 28 10:49 Dans_ma_bull
-rw-r--r-- 1 hacker hacker 1.4M Jun 28 10:49 Flora_by_Mar
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 Icy_Grass_by
-rw-r--r-- 1 hacker hacker 1.1M Jun 28 10:49 Night_lights
-rw-rw-r-- 1 hacker hacker 3.4K Jun 28 10:52 nr.jpg
-rw-r--r-- 1 hacker hacker 834K Jun 28 10:49 passion_flow
-rw-r--r-- 1 hacker hacker 1.5M Jun 28 10:49 Spring_by_Peter_Apas.jpg
-rw-r--r-- 1 hacker hacker 1.7M Jun 28 10:49 TCP118v1_by_Tiziano_Consonni.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
-rw-rw-r-- 1 hacker hacker 180K Jun 28 10:50 warty-final-ubuntu.jpg
-rw-r--r-- 1 hacker hacker 2.6M Jun 28 10:49 warty-final-ubuntu.png
-rw-rw-r-- 1 hacker hacker 2.5K Jun 28 10:52 warty.jpg
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:52:59 AM] [~/course/backgrounds]
-> % identify nr.jpg
nr.jpg JPEG 126x80 126x80+0+0 8-bit Gray 256c 3.44KB 0.000u 0:00.000
hacker@laptop [10:53:14 AM] [~/course/backgrounds]
-> % eog nr.jpg
```

- Now let's look at some image shrinking tools outside of `imagemagick`. The first one is a `png` shrinking tool called `pngquant`. We will install it by typing the following:
  `sudo apt install pngquant`



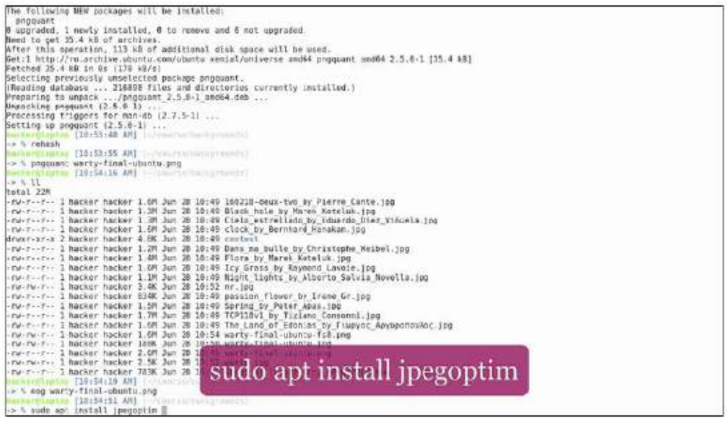
```
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:50:41 AM] [~/course/backgrounds]
-> % identify warty-final-ubuntu.jpg
warty-final-ubuntu.jpg JPEG 4096x2304 4096x2304+0+0 8-bit sRGB 184KB 0.000u 0:00.009
hacker@laptop [10:50:57 AM] [~/course/backgrounds]
-> % eog warty-final-ubuntu.png
hacker@laptop [10:51:21 AM] [~/course/backgrounds]
-> % convert -crop "500x500+100+100" warty-final-ubuntu.jpg warty.jpg
hacker@laptop [10:52:03 AM] [~/course/backgrounds]
-> % identify warty.jpg
warty.jpg JPEG 500x500 500x500+0+0 8-bit sRGB 2.5KB 0.000u 0:00.000
hacker@laptop [10:52:11 AM] [~/course/backgrounds]
-> % eog warty.jpg
hacker@laptop [10:52:32 AM] [~/course/backgrounds]
-> % convert -size x80 label:123 nr.jpg
hacker@laptop [10:52:56 AM] [~/course/backgrounds]
-> % ll
total 20M
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 160218-deux-two_by_Pierre_Cante.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Black_hole_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Cielo_estrellado_by_Eduardo_Diez_Viñuela.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 clock_by_Bernhard_Hanakam.jpg
drwxr-xr-x 2 hacker hacker 4.0K Jun 28 10:49 contest
-rw-r--r-- 1 hacker hacker 1.2M Jun 28 10:49 Dans_ma_bulle_by_Christophe_Weibel.jpg
-rw-r--r-- 1 hacker hacker 1.4M Jun 28 10:49 Flora_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 Icy_Grass_by_Raymond_Lavoie.jpg
-rw-r--r-- 1 hacker hacker 1.1M Jun 28 10:49 Night_lights_by_Alberto_Salvia_Novella.jpg
-rw-rw-r-- 1 hacker hacker 3.4K Jun 28 10:52 nr.jpg
-rw-r--r-- 1 hacker hacker 834K Jun 28 10:49 passion_flower_by_Irene_Gr.jpg
-rw-r--r-- 1 hacker hacker 1.5M Jun 28 10:49 Spring_by_Peter_Apas.jpg
-rw-r--r-- 1 hacker hacker 1.7M Jun 28 10:49 TCP118v1_by_Tiziano_Consonni.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
-rw-rw-r-- 1 hacker hacker 180K Jun 28 10:50 warty-final-ubuntu.jpg
-rw-r--r-- 1 hacker hacker 2.6M Jun 28 10:49 warty-final-ubuntu.png
-rw-rw-r-- 1 hacker hacker 2.5K Jun 28 10:52 warty.jpg
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:52:59 AM] [~/course/backgrounds]
-> % identify nr.jpg
nr.jpg JPEG 126x80 126x80+0+0 8-bit Gray
hacker@laptop [10:53:14 AM] [~/course/backgrounds]
-> % eog nr.jpg
hacker@laptop [10:53:28 AM] [~/course/backgrounds]
-> % sudo apt install pngquant
```

- To shrink the large PNG imagewe would just call `pngquant` with the following image name:
  `pngquant warty-final-ubuntu.png`



```
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 clock_by_Bernhard_Hanakam.jpg
drwxr-xr-x 2 hacker hacker 4.0K Jun 28 10:49 contest
-rw-r--r-- 1 hacker hacker 1.2M Jun 28 10:49 Dans_ma_bulle_by_Christophe_Weibel.jpg
-rw-r--r-- 1 hacker hacker 1.4M Jun 28 10:49 Flora_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 Icy_Grass_by_Raymond_Lavoie.jpg
-rw-r--r-- 1 hacker hacker 1.1M Jun 28 10:49 Night_lights_by_Alberto_Salvia_Novella.jpg
-rw-rw-r-- 1 hacker hacker 3.4K Jun 28 10:52 nr.jpg
-rw-r--r-- 1 hacker hacker 834K Jun 28 10:49 passion_flower_by_Irene_Gr.jpg
-rw-r--r-- 1 hacker hacker 1.5M Jun 28 10:49 Spring_by_Peter_Apas.jpg
-rw-r--r-- 1 hacker hacker 1.7M Jun 28 10:49 TCP118v1_by_Tiziano_Consonni.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
-rw-rw-r-- 1 hacker hacker 180K Jun 28 10:50 warty-final-ubuntu.jpg
-rw-r--r-- 1 hacker hacker 2.6M Jun 28 10:49 warty-final-ubuntu.png
-rw-rw-r-- 1 hacker hacker 2.5K Jun 28 10:52 warty.jpg
-rw-r--r-- 1 hacker hacker 783K Jun 28 10:49 Xerus_Wallpaper_Grey_4096x2304.png
hacker@laptop [10:52:59 AM] [~/course/backgrounds]
-> % identify nr.jpg
nr.jpg JPEG 126x80 126x80+0+0 8-bit Gray 256c 3.44KB 0.000u 0:00.000
hacker@laptop [10:53:14 AM] [~/course/backgrounds]
-> % eog nr.jpg
hacker@laptop [10:53:28 AM] [~/course/backgrounds]
-> % sudo apt install pngquant
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  pngquant
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 35.4 kB of archives.
After this operation, 113 kB of additional disk space will be used.
Get:1 http://ro.archive.ubuntu.com/ubuntu xenial/universe amd64 pngquant amd64 2.5.0-1 [35.4 kB]
Fetched 35.4 kB in 0s (178 kB/s)
Selecting previously unselected package pngquant.
(Reading database ... 216098 files and directories currently installed.)
Preparing to unpack .../pngquant_2.5.0-1_amd64.deb ...
Unpacking pngquant (2.5.0-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up pngquant (2.5.0-1) ...
hacker@laptop [10:53:48 AM] [~/course/backgrounds]
-> % rehash
hacker@laptop [10:53:55 AM] [~/course/backgrounds]
-> % pngquant warty-final-ubuntu.png
```

- To shrink the JPEG image, we'll install the equivalent of `pngquant`, which is `jpegoptim`:
  `sudo apt install jpegoptim`



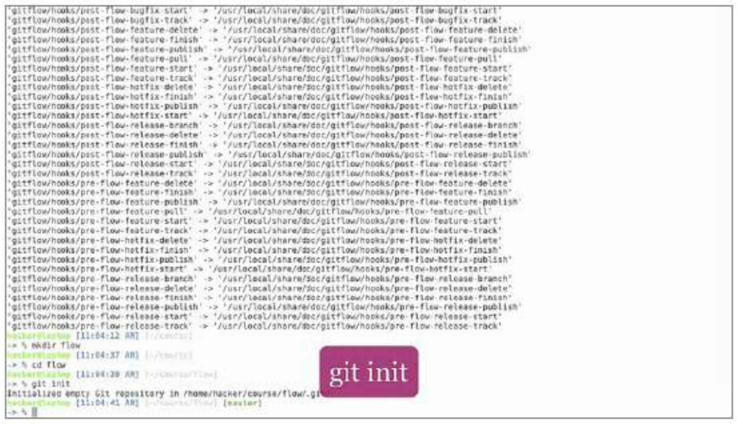
```
The following NEW packages will be installed:
  pngquant
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 35.4 kB of archives.
After this operation, 113 kB of additional disk space will be used.
Get:1 http://ro.archive.ubuntu.com/ubuntu xenial/universe amd64 pngquant amd64 2.5.0-1 [35.4 kB]
Fetched 35.4 kB in 0s (178 kB/s)
Selecting previously unselected package pngquant.
(Reading database ... 216898 files and directories currently installed.)
Preparing to unpack .../pngquant_2.5.0-1_amd64.deb ...
Unpacking pngquant (2.5.0-1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up pngquant (2.5.0-1) ...
hacker@laptop [10:53:48 AM] [~/course/backgrounds]
-> % rehash
hacker@laptop [10:53:55 AM] [~/course/backgrounds]
-> % pngquant warty-final-ubuntu.png
hacker@laptop [10:54:16 AM] [~/course/backgrounds]
-> % ll
total 22M
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 160218-deux-two_by_Pierre_Cante.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Black_hole_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.3M Jun 28 10:49 Cielo_estrellado_by_Eduardo_Diez_Viñuela.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 clock_by_Bernhard_Hanakam.jpg
drwxr-xr-x 2 hacker hacker 4.0K Jun 28 10:49 contest
-rw-r--r-- 1 hacker hacker 1.2M Jun 28 10:49 Dans_ma_bulle_by_Christophe_Weibel.jpg
-rw-r--r-- 1 hacker hacker 1.4M Jun 28 10:49 Flora_by_Marek_Koteluk.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 Icy_Grass_by_Raymond_Lavoie.jpg
-rw-r--r-- 1 hacker hacker 1.1M Jun 28 10:49 Night_lights_by_Alberto_Salvia_Novella.jpg
-rw-rw-r-- 1 hacker hacker 3.4K Jun 28 10:52 nr.jpg
-rw-r--r-- 1 hacker hacker 834K Jun 28 10:49 passion_flower_by_Irene_Gr.jpg
-rw-r--r-- 1 hacker hacker 1.5M Jun 28 10:49 Spring_by_Peter_Apas.jpg
-rw-r--r-- 1 hacker hacker 1.7M Jun 28 10:49 TCP118v1_by_Tiziano_Consonni.jpg
-rw-r--r-- 1 hacker hacker 1.6M Jun 28 10:49 The_Land_of_Edonias_by_Γιωργος_Αργυροπουλος.jpg
-rw-rw-r-- 1 hacker hacker 1.6M Jun 28 10:54 warty-final-ubuntu-fs8.png
-rw-rw-r-- 1 hacker hacker 186K Jun 28 10:5
-rw-r--r-- 1 hacker hacker 2.0M Jun 28 1
-rw-rw-r-- 1 hacker hacker 2.5K Jun 28 1
-rw-r--r-- 1 hacker hacker 783K Jun 28 1
hacker@laptop [10:54:19 AM] [~/course/b
-> % eog warty-final-ubuntu.png
hacker@laptop [10:54:51 AM] [~/course/backgrounds]
-> % sudo apt install jpegoptim
```

Go with the Git flow

Git is by far the most popular version control system out there.

- We will be looking at a variation of the `git-flow` plugin, called `gitflowavh`, which adds extra functionality, such as **Git hooks**, `https://github.com/petervanderdoes/gitflow-avh`.
- Once this is done, let's create a dummy project. We'll create an empty directory and initialize it as a Git repository:

  `git init`



```
'gitflow/hooks/post-flow-bugfix-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-start'
'gitflow/hooks/post-flow-bugfix-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-bugfix-track'
'gitflow/hooks/post-flow-feature-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-delete'
'gitflow/hooks/post-flow-feature-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-finish'
'gitflow/hooks/post-flow-feature-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-publish'
'gitflow/hooks/post-flow-feature-pull' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-pull'
'gitflow/hooks/post-flow-feature-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-start'
'gitflow/hooks/post-flow-feature-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-feature-track'
'gitflow/hooks/post-flow-hotfix-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-delete'
'gitflow/hooks/post-flow-hotfix-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-finish'
'gitflow/hooks/post-flow-hotfix-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-publish'
'gitflow/hooks/post-flow-hotfix-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-hotfix-start'
'gitflow/hooks/post-flow-release-branch' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-branch'
'gitflow/hooks/post-flow-release-delete' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-delete'
'gitflow/hooks/post-flow-release-finish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-finish'
'gitflow/hooks/post-flow-release-publish' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-publish'
'gitflow/hooks/post-flow-release-start' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-start'
'gitflow/hooks/post-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/post-flow-release-track'
'gitflow/hooks/pre-flow-feature-delete' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-delete'
'gitflow/hooks/pre-flow-feature-finish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-finish'
'gitflow/hooks/pre-flow-feature-publish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-publish'
'gitflow/hooks/pre-flow-feature-pull' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-pull'
'gitflow/hooks/pre-flow-feature-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-start'
'gitflow/hooks/pre-flow-feature-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-feature-track'
'gitflow/hooks/pre-flow-hotfix-delete' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-delete'
'gitflow/hooks/pre-flow-hotfix-finish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-finish'
'gitflow/hooks/pre-flow-hotfix-publish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-publish'
'gitflow/hooks/pre-flow-hotfix-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-hotfix-start'
'gitflow/hooks/pre-flow-release-branch' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-branch'
'gitflow/hooks/pre-flow-release-delete' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-delete'
'gitflow/hooks/pre-flow-release-finish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-finish'
'gitflow/hooks/pre-flow-release-publish' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-publish'
'gitflow/hooks/pre-flow-release-start' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-start'
'gitflow/hooks/pre-flow-release-track' -> '/usr/local/share/doc/gitflow/hooks/pre-flow-release-track'
hacker@laptop [11:04:12 AM] [~/course]
-> % mkdir flow
hacker@laptop [11:04:37 AM] [~/course]
-> % cd flow
hacker@laptop [11:04:39 AM] [~/course/flow]
-> % git init
Initialized empty Git repository in /home/hacker/course/flow/.gi
hacker@laptop [11:04:41 AM] [~/course/flow] [master]
-> %
```

- http://danielkummer.github.io/git-flow-cheatsheet/ provides the basic tips and tricks to get you started quickly with `git-flow`.
- The first thing the cheatsheet suggests is to run the following:

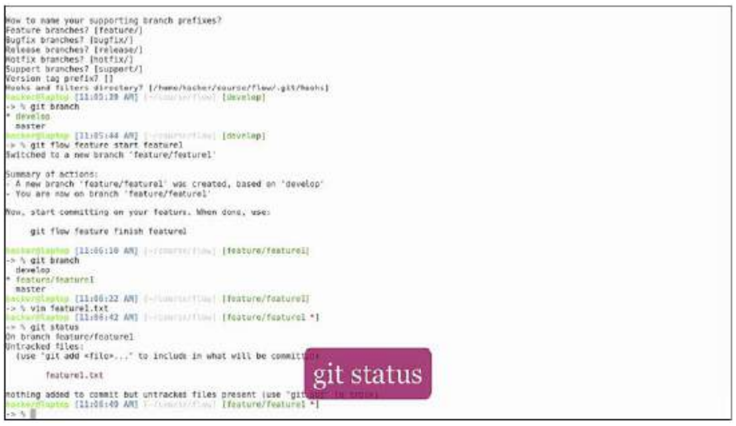
  `git flow init`
- To work the develop branch, we can run

  `git branch`
- Feature branches are useful when developing a specific part of functionalityor doing refactoring, but you don't want to break the existing functionality on the develop branch. To create a feature branch, just run the following:

  `git flow feature start feature1`

`GitFlow` will also show us a summary of actions once the feature branch has finished. This has created a new branch called `feature/ feature1`, based on the develop branch and has switched us to that branch. We can also see this from our handy `zsh` prompt.

- Let's open up a file, edit, and save it:
  `git status`



```
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/hacker/source/flow/.git/hooks]
hacker@laptop [11:03:39 AM] [~/source/flow] [develop]
-> % git branch
* develop
  master
hacker@laptop [11:05:44 AM] [~/source/flow] [develop]
-> % git flow feature start feature1
Switched to a new branch 'feature/feature1'

Summary of actions:
- A new branch 'feature/feature1' was created, based on 'develop'
- You are now on branch 'feature/feature1'

Now, start committing on your feature. When done, use:

     git flow feature finish feature1

hacker@laptop [11:06:10 AM] [~/source/flow] [feature/feature1]
-> % git branch
  develop
* feature/feature1
  master
hacker@laptop [11:06:22 AM] [~/source/flow] [feature/feature1]
-> % vim feature1.txt
hacker@laptop [11:06:42 AM] [~/source/flow] [feature/feature1 *]
-> % git status
On branch feature/feature1
Untracked files:
  (use "git add <file>..." to include in what will be committ

        feature1.txt

nothing added to commit but untracked files present (use "git
hacker@laptop [11:06:49 AM] [~/source/flow] [feature/feature1 *]
-> %
```

Now `git commit` is using the `nano` editor for editing the commit message. Since we prefer `vim`, let's go ahead and change the default editor to `vim`. All we need to do is add this line in our `zshrc` and reload it:
`export EDITOR=vim`

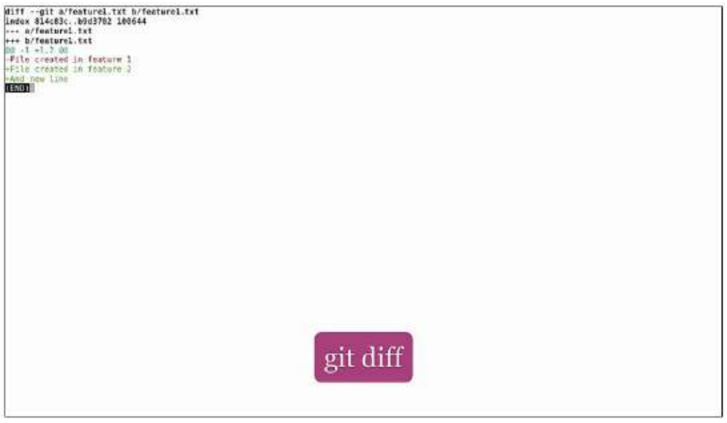
- Now when we do a `git commit` Vim opens up, shows us a summary of the commit, and closes.
- It's time to merge the feature branch back to develop with the following:
  `git flow feature finish feature1`
- To develope release branch
  `git flow release start 1.0.0`
- To finish our release branch
  `git flow release finish 1.0.0`

- To develope branch
  `git branch`
- To see that the only two available branches are master and develop:
  `git tag`
- To check whether the branch is correct
  `git flow hotfix start 1.0.1`
- To delete our branch:
  `git flow hotfix delete 1.0.1`
- To finish the hotflix
  `git flow hotfix finish 1.0.1`

Merging Git Conflict with Ease

- Let's open the `feature` file from our previous chapter, edit it, add a new line, and save it:
  `git diff`

- The `git diff` command will show us colored text explaining the differences between the `git` file and the modified file, but some people find this format hard to understand:



```
diff --git a/feature1.txt b/feature1.txt
index 814c83c..b9d3782 100644
--- a/feature1.txt
+++ b/feature1.txt
@@ -1 +1,2 @@
-File created in feature 1
+File created in feature 2
+And new line
(END)
```

- Let's install it using the following:
  `sudo apt install meld`
- To launch Meld
  `git difftool`
- For resolving the conflict, we will be using the following command:
  `git mergetool`
- To add `of text`:
  `git commit -a`
-

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : David Agustriawan
Email : david.agustriawan@i3l.ac.id

Supervisor(s)	email(s)
Arli Aditya Parikesit	arli.parikesit@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
    u. If you have any troubleshoot please contact your supervisor or Building Management
    v. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
    w. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
    x. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 6

Date : October 8th, 2019

Laboratory :Bioinformatics laboratory

Overview

Terminal Art is the part of Terminal that can create something creative and fun to the Terminal. Terminals can be stylish and can give a good impression, especially the ones we see in the movies. Colors, ASCII art, and animations can make our terminal come alive.

Objective

Students are expected to understand on how applying Linux development tools.

Procedure

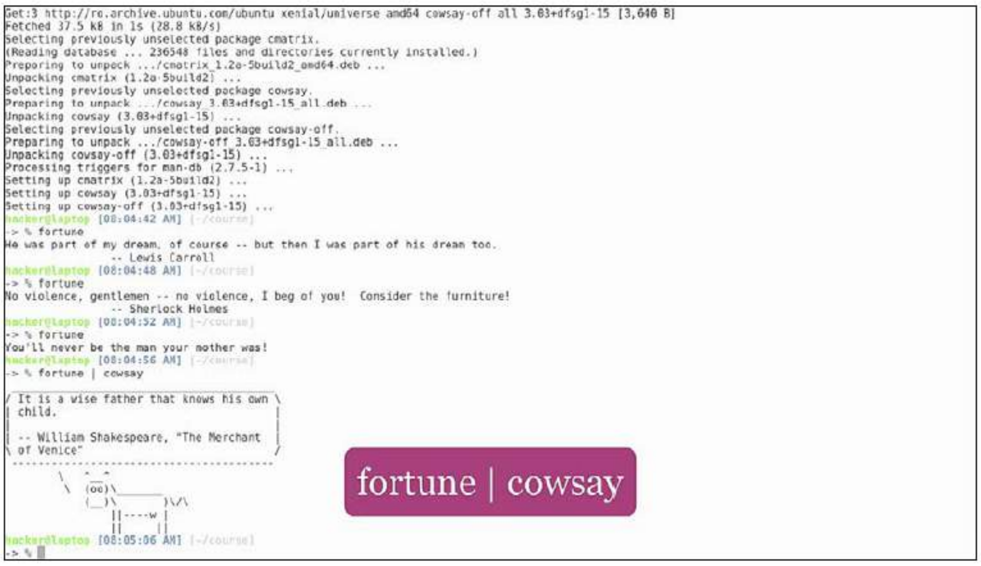
- First of all, the think we need to do is installing the utilities that we will be using for creating a wonderfull Terminal.

  ```
  sudo apt install fortune cowsay cmatrix
  ```
- Then run this command

  ```
  fortune
  ```

  When running this command, you get fortunes, quotes, and jokes, in a random order
- If we combine the command with cowsay, we will get the same fortunes, delivered with an image of a cow:

  ```
  fortune | cowsay
  ```



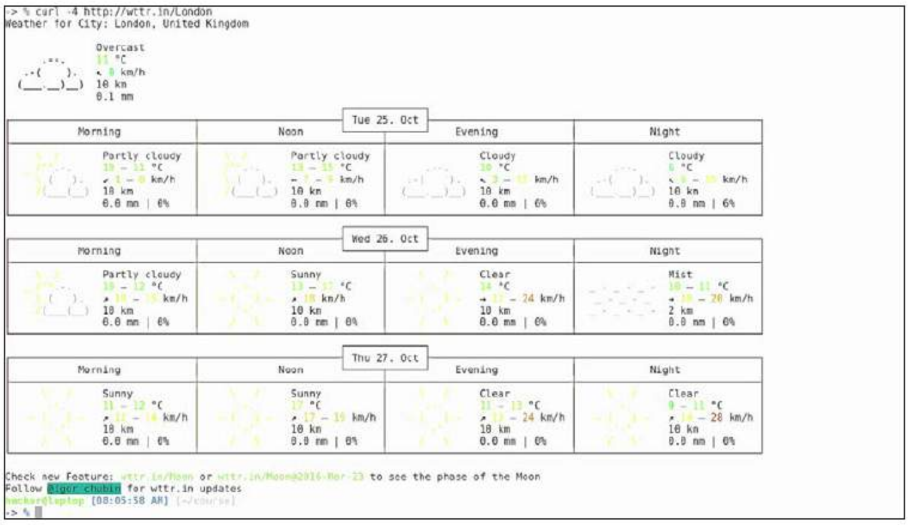
```
Get:3 http://ro.archive.ubuntu.com/ubuntu xenial/universe amd64 cowsay-off all 3.03+dfsg1-15 [3,640 B]
Fetched 37.5 kB in 1s (28.8 kB/s)
Selecting previously unselected package cmatrix.
(Reading database ... 236548 files and directories currently installed.)
Preparing to unpack .../cmatrix_1.2a-5build2_amd64.deb ...
Unpacking cmatrix (1.2a-5build2) ...
Selecting previously unselected package cowsay.
Preparing to unpack .../cowsay_3.03+dfsg1-15_all.deb ...
Unpacking cowsay (3.03+dfsg1-15) ...
Selecting previously unselected package cowsay-off.
Preparing to unpack .../cowsay-off_3.03+dfsg1-15_all.deb ...
Unpacking cowsay-off (3.03+dfsg1-15) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up cmatrix (1.2a-5build2) ...
Setting up cowsay (3.03+dfsg1-15) ...
Setting up cowsay-off (3.03+dfsg1-15) ...
hacker@laptop [08:04:42 AM] [~/course]
-> % fortune
He was part of my dream, of course -- but then I was part of his dream too.
                -- Lewis Carroll
hacker@laptop [08:04:48 AM] [~/course]
-> % fortune
No violence, gentlemen -- no violence, I beg of you!  Consider the furniture!
                -- Sherlock Holmes
hacker@laptop [08:04:52 AM] [~/course]
-> % fortune
You'll never be the man your mother was!
hacker@laptop [08:04:56 AM] [~/course]
-> % fortune | cowsay
 _________________________________________
/ It is a wise father that knows his own \
| child.                                  |
|                                         |
| -- William Shakespeare, "The Merchant   |
\ of Venice"                              /
 -----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
hacker@laptop [08:05:06 AM] [~/course]
-> %
```

To make this recurrent, we can include it as the last line in our `zshrc` file. Then, every time we open a new terminal window, a cow will deliver a fortune to us.

- To predict the weather, we can write curl command

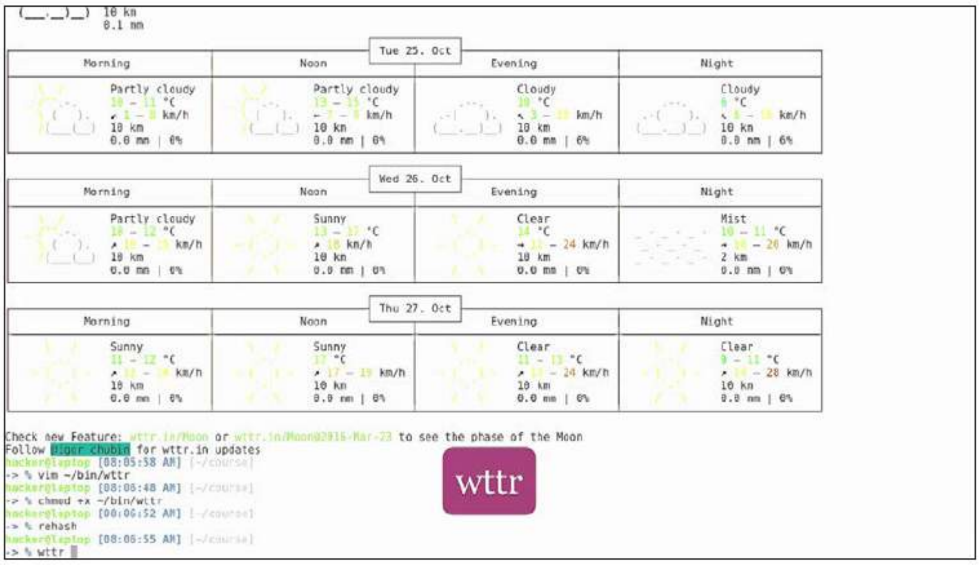
  ```
  curl -4 http://wttr.in/citynames
  ```

This will show, in a nice format, a three-day weather forecast for the specified city, in this case, London:

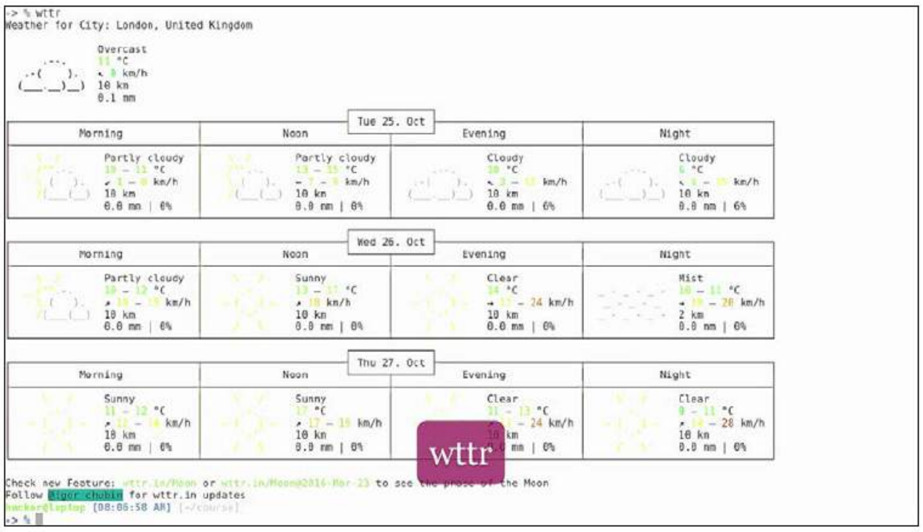


```
-> % curl -4 http://wttr.in/London
Weather for City: London, United Kingdom

                Overcast
                11 °C
                ↖ 8 km/h
                10 km
                0.1 mm

                                  Tue 25. Oct
      Morning            Noon               Evening            Night
 Partly cloudy      Partly cloudy      Cloudy             Cloudy
 10 – 11 °C         13 – 15 °C         10 °C              8 °C
 ↙ 1 – 0 km/h       ← 7 – 9 km/h       ↖ 3 – 12 km/h      ↖ 8 – 10 km/h
 10 km              10 km              10 km              10 km
 0.0 mm | 6%        0.0 mm | 0%        0.0 mm | 6%        0.0 mm | 6%

                                  Wed 26. Oct
      Morning            Noon               Evening            Night
 Partly cloudy      Sunny              Clear              Mist
 10 – 12 °C         13 – 17 °C         14 °C              10 – 11 °C
 ↗ 10 – 15 km/h     ↗ 18 km/h          → 17 – 24 km/h     → 18 – 20 km/h
 10 km              10 km              10 km              2 km
 0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%

                                  Thu 27. Oct
      Morning            Noon               Evening            Night
 Sunny              Sunny              Clear              Clear
 11 – 12 °C         17 °C              11 – 13 °C         9 – 11 °C
 ↗ 11 – 14 km/h     ↗ 17 – 19 km/h     ↗ 19 – 24 km/h     ↗ 14 – 28 km/h
 10 km              10 km              10 km              10 km
 0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%

Check new Feature: wttr.in/Moon or wttr.in/Moon@2016-Mar-23 to see the phase of the Moon
Follow @igor_chubin for wttr.in updates
hacker@laptop [08:05:58 AM] [~/course]
-> %
```

- Now, with our newly learned skills, let's put together a shell script that gives us the weather forecast:
  Open ~/bin/wttr and type the following:
  `#!/bin/bash`
  `CITY=${1:-London}`
  `curl -4 http://wttr.in/${CITY}`
  Give it execution rights and assign a default city, let's say London. Now, run this:
  `wttr`



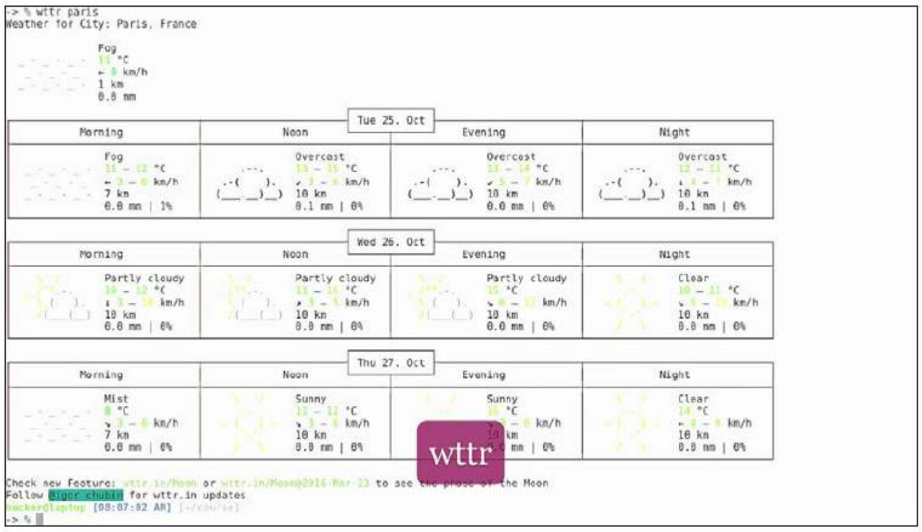
```
                10 km
                0.1 mm

                                  Tue 25. Oct
      Morning            Noon               Evening            Night
 Partly cloudy      Partly cloudy      Cloudy             Cloudy
 10 – 11 °C         13 – 15 °C         10 °C              8 °C
 ↙ 1 – 0 km/h       ← 7 – 9 km/h       ↖ 3 – 12 km/h      ↖ 8 – 10 km/h
 10 km              10 km              10 km              10 km
 0.0 mm | 6%        0.0 mm | 0%        0.0 mm | 6%        0.0 mm | 6%

                                  Wed 26. Oct
      Morning            Noon               Evening            Night
 Partly cloudy      Sunny              Clear              Mist
 10 – 12 °C         13 – 17 °C         14 °C              10 – 11 °C
 ↗ 10 – 15 km/h     ↗ 18 km/h          → 17 – 24 km/h     → 18 – 20 km/h
 10 km              10 km              10 km              2 km
 0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%

                                  Thu 27. Oct
      Morning            Noon               Evening            Night
 Sunny              Sunny              Clear              Clear
 11 – 12 °C         17 °C              11 – 13 °C         9 – 11 °C
 ↗ 11 – 14 km/h     ↗ 17 – 19 km/h     ↗ 19 – 24 km/h     ↗ 14 – 28 km/h
 10 km              10 km              10 km              10 km
 0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%        0.0 mm | 0%

Check new Feature: wttr.in/Moon or wttr.in/Moon@2016-Mar-23 to see the phase of the Moon
Follow @igor_chubin for wttr.in updates
hacker@laptop [08:05:58 AM] [~/course]
-> % vim ~/bin/wttr
hacker@laptop [08:06:48 AM] [~/course]
-> % chmod +x ~/bin/wttr
hacker@laptop [08:06:52 AM] [~/course]
-> % rehash
hacker@laptop [08:06:55 AM] [~/course]
-> % wttr
```



```
-> % wttr
Weather for City: London, United Kingdom

       Overcast
       11 °C
       ↖ 8 km/h
       10 km
       0.1 mm

                                   Tue 25. Oct
Morning               Noon                  Evening               Night
Partly cloudy         Partly cloudy         Cloudy                Cloudy
10 – 11 °C            13 – 15 °C            10 °C                 6 °C
↙ 1 – 6 km/h          ← 7 – 9 km/h          ↖ 3 – 13 km/h         ↖ 8 – 15 km/h
10 km                 10 km                 10 km                 10 km
0.0 mm | 0%           0.0 mm | 0%           0.0 mm | 6%           0.0 mm | 6%

                                   Wed 26. Oct
Morning               Noon                  Evening               Night
Partly cloudy         Sunny                 Clear                 Mist
10 – 12 °C            13 – 17 °C            14 °C                 10 – 11 °C
↗ 10 – 15 km/h        ↗ 18 km/h             → 17 – 24 km/h        → 19 – 20 km/h
10 km                 10 km                 10 km                 2 km
0.0 mm | 0%           0.0 mm | 0%           0.0 mm | 0%           0.0 mm | 0%

                                   Thu 27. Oct
Morning               Noon                  Evening               Night
Sunny                 Sunny                 Clear                 Clear
11 – 12 °C            17 °C                 11 – 13 °C            9 – 11 °C
↗ 12 – 14 km/h        ↗ 17 – 19 km/h          – 24 km/h           ↗ 14 – 28 km/h
10 km                 10 km                    km                 10 km
0.0 mm | 0%           0.0 mm | 0%              mm | 0%            0.0 mm | 0%

Check new Feature: wttr.in/Moon or wttr.in/Moon@2016-Mar-23 to see the phase of the Moon
Follow @igor_chubin for wttr.in updates
hacker@laptop [08:06:58 AM] [~/course]
-> %
```

These figures show the wheater in London since we put “London” in the command

- We get the weather forecast for London. Now, run this:
  `wttr paris`



```
-> % wttr paris
Weather for City: Paris, France

       Fog
       11 °C
       ← 8 km/h
       1 km
       0.0 mm

                                   Tue 25. Oct
Morning               Noon                  Evening               Night
Fog                   Overcast              Overcast              Overcast
11 – 12 °C            13 – 15 °C            13 – 14 °C            12 – 13 °C
← 3 – 6 km/h          ↙ 3 – 6 km/h          ↙ 5 – 7 km/h          ↓ 4 – 7 km/h
7 km                  10 km                 10 km                 10 km
0.0 mm | 1%           0.1 mm | 0%           0.0 mm | 0%           0.1 mm | 0%

                                   Wed 26. Oct
Morning               Noon                  Evening               Night
Partly cloudy         Partly cloudy         Partly cloudy         Clear
10 – 12 °C            11 – 16 °C            15 °C                 10 – 11 °C
↓ 3 – 10 km/h         ↗ 3 – 5 km/h          ↘ 6 – 12 km/h         ↘ 6 – 10 km/h
10 km                 10 km                 10 km                 10 km
0.0 mm | 0%           0.0 mm | 0%           0.0 mm | 0%           0.0 mm | 0%

                                   Thu 27. Oct
Morning               Noon                  Evening               Night
Mist                  Sunny                 Sunny                 Clear
8 °C                  11 – 12 °C            16 °C                 14 °C
↘ 3 – 6 km/h          ↘ 3 – 6 km/h            – 6 km/h            ← 4 – 8 km/h
7 km                  10 km                    km                 10 km
0.0 mm | 0%           0.0 mm | 0%              mm | 0%            0.0 mm | 0%

Check new Feature: wttr.in/Moon or wttr.in/Moon@2016-Mar-23 to see the phase of the Moon
Follow @igor_chubin for wttr.in updates
hacker@laptop [08:07:02 AM] [~/course]
-> %
```

- To create matrix or amazing output. Type these command
  `cmatrix`

Indonesia International Institute for Life Science

[BI2107]

LINUX AND R COMPUTATION

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer	: Nanda R. Pradana Ratnasari
Email	: nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   y. If you have any troubleshoot please contact your supervisor or Building Management
   z. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   aa. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   bb. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 7

Date : October 15th, 2019

Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

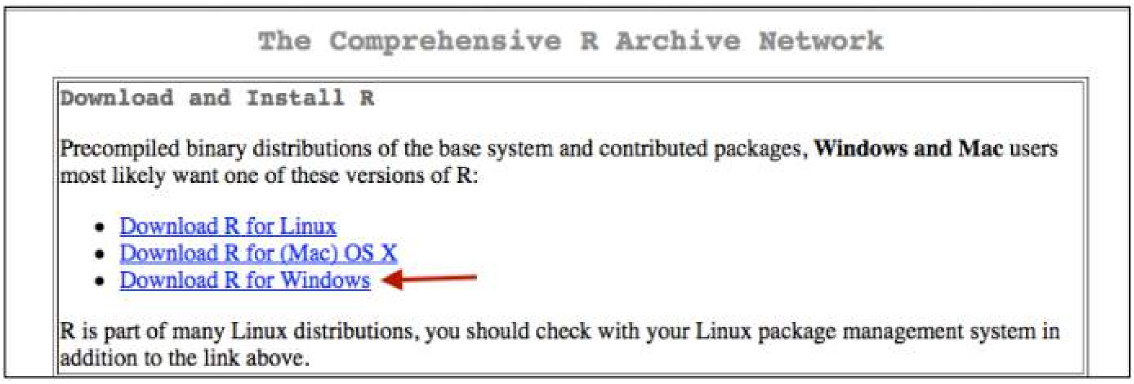
- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

Procedure

First click on https://www.r-project.org/ then "Download R for Windows" then click on "install R for the first time" and then click on "Download R 3.1.3 for Windows" (or the more current version). These clicks are illustrated in the screenshots below show and the arrows point to the location of each click.



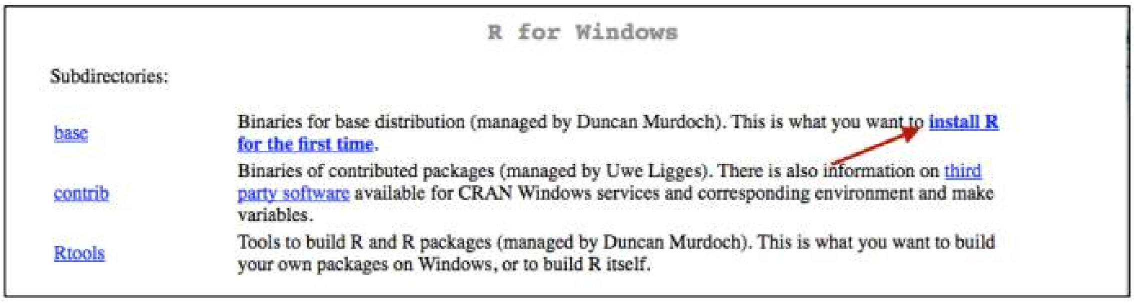
The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

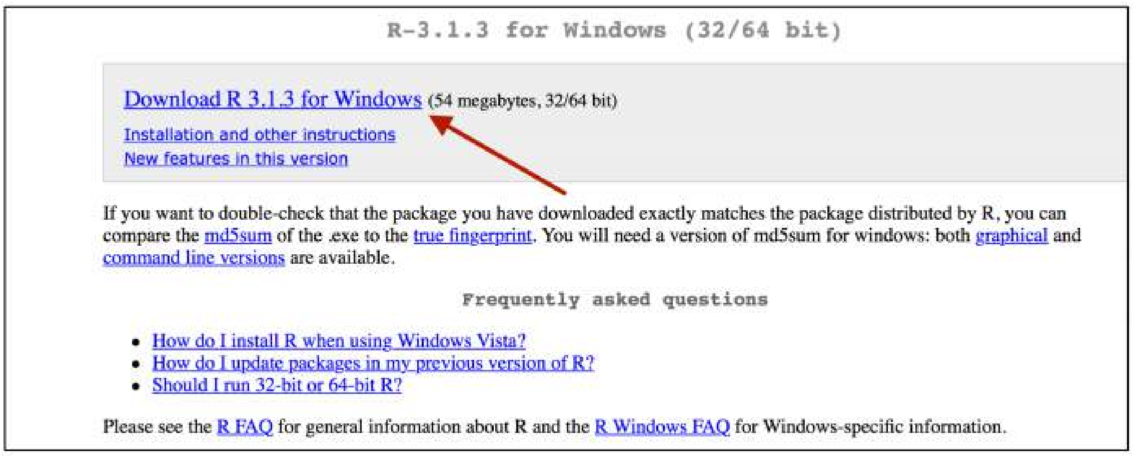
- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.


R for Windows

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to install R for the first time.
contrib	Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.


R-3.1.3 for Windows (32/64 bit)

Download R 3.1.3 for Windows (54 megabytes, 32/64 bit)

Installation and other instructions

New features in this version

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the md5sum of the .exe to the true fingerprint. You will need a version of md5sum for windows: both graphical and command line versions are available.

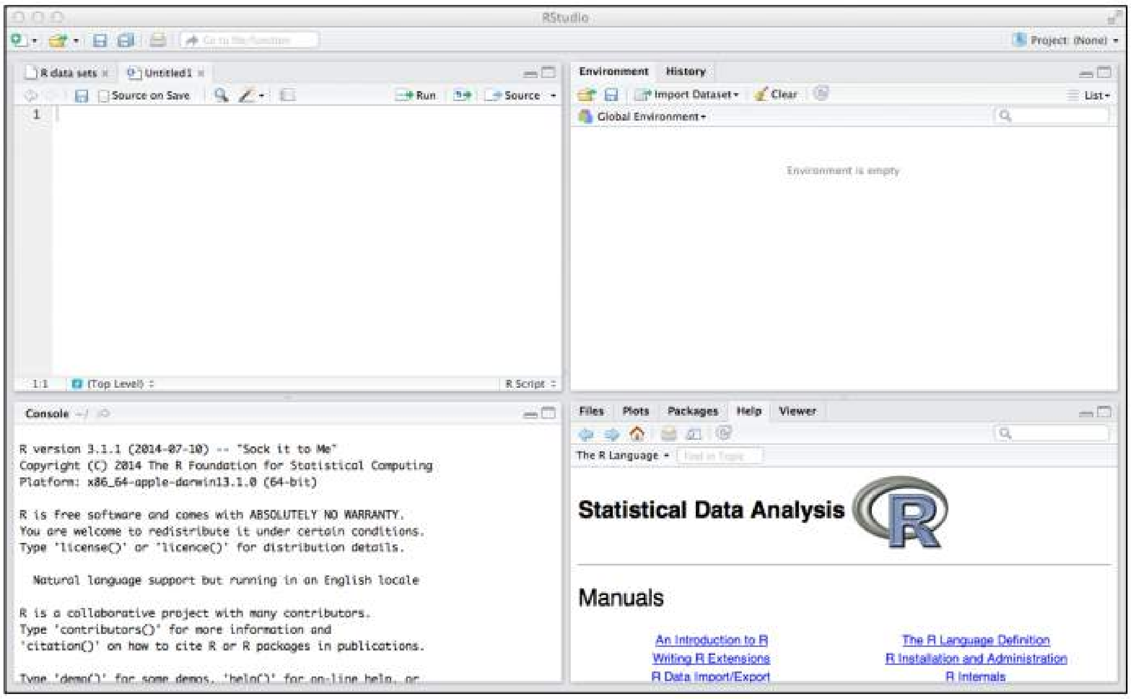
Frequently asked questions

- How do I install R when using Windows Vista?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

A. The 4 Panes

Welcome to RStudio. Below is a screenshot of the RStudio window. The four panes might be arranged in a different order on your computer, but you should see something similar.



```
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.1.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
```

1. The console pane is where the action takes place (located lower left in the example above). From this screenshot you see that this session was running R version 3.1.1 called "Sock it to Me," which was released July 10, 2014. The console is the heart of RStudio. You can type commands directly into the console whenever you see the flashing cursor. Output and error messages are displayed in the console.
2. The script or source pane (located upper left in the example above) is where you can type and save your commands and make notes to yourself about projects. When you run a command from the source pane, the command is sent over to the console pane to be executed. It is possible to have multiple sources or scripts appear in the source pane, and they will each have their own tab at the top of the pane. More on this topic later in the Lab.
3. The environment and history pane (located upper right in the example above) is where you will see the different objects you create or the different datasets you import.
4. The final pane contains everything else including help, plots, packages, etc (located in the lower right in the example above). You will become more familiar with all of these panes as you move through the Lab.

B. Calculator in RStudio

One of the more simple uses of RStudio is to use it like a calculator:

1. Locate the flashing cursor in the console pane.

2. Type basic addition (+), subtraction (-), multiplication (*), and division procedures (/) directly into the console pane. RStudio is mostly flexible about using spaces, so you can include spaces between the characters or not.

3. Hit enter/return after each command.

4. RStudio will perform the mathematic procedure and return the result in the console in the line below next to a [1] that indicates the first and, in this case, only result from your command.

5. Try these examples or variations of these examples:

```
6+6
```

```
75-25
12 * 3
180 / 12
```

6. Try more advanced calculator examples:

```
17.333 * 0.72
10 + 10 * 10
(10 + 10) * 10
10^3
```

C. Text & Error Messages in RStudio

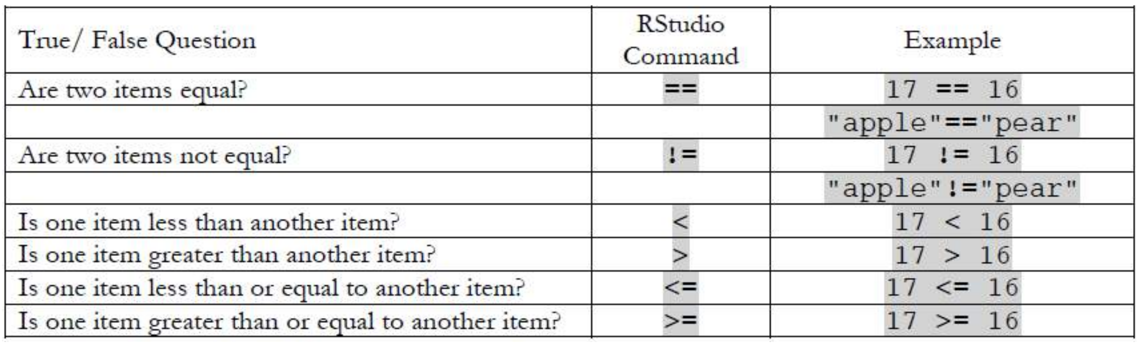
Text is often referred to as “strings” in programming and in statistics. You will be using text in RStudio, but in this task below we will use text to see what error messages look like. Text (strings) requires quotation marks ("") to tell RStudio that the text is different than a command or function.

1. Type a word or words between quotation marks directly into the console pane.
   For example: "hello world"
   RStudio will print the words in the line below the command as a result.
2. Notice the error message when text is used in the basic calculator commands. Type:
   ```
   10 – "one"
   5 + "apple"
   ```

This might be your first error message in RStudio. Notice that there are some details included to help diagnose the error. Sometimes the error messages can be difficult to understand. The error message for trying to add the word "apple" to the number 5 tells us that "apple" is nonnumeric. Nothing bad has happened in RStudio. This error means that RStudio wasn’t able to complete the command. Errors signal that you should stop and check your command and that it might not be appropriate to move forward to the next command until the error is resolved.

D. True/ False Questions in RStudio

You can ask RStudio true/false questions.



True/ False Question	RStudio Command	Example
Are two items equal?	`==`	`17 == 16`
		`"apple"=="pear"`
Are two items not equal?	`!=`	`17 != 16`
		`"apple"!="pear"`
Is one item less than another item?	`<`	`17 < 16`
Is one item greater than another item?	`>`	`17 > 16`
Is one item less than or equal to another item?	`<=`	`17 <= 16`
Is one item greater than or equal to another item?	`>=`	`17 >= 16`

1. Type each of the true/false examples from the table above in the console pane. Note that you will get errors if you include spaces within these commands. For example, the command for equals to is == not = =. Space between the numbers and the commands (see example in the table above) is a matter of preference.

These examples might seem like silly true/ false questions since we know the answers just by comparing the two values. However, imagine that 17 was a variable containing the number of arrests of one individual and 16 was a different variable containing the number of arrests of a second individual. Once we move into objects, these true/false questions become more useful.

E. Creating and Modifying Objects in Rstudio

As you move through the modules you will become familiar with the assign command (<-) in RStudio. This little arrow assigns everything on the right of the arrow to the item specified on the left of the arrow.

1. Create a new object called a and assign it the value 15. Then create an object called b and assign it the value 20:

   ```
   a <- 15
   b <- 20
   ```

2. Notice what happens in the environment pane in the upper right. Because these two commands created new objects called a and b, these new objects appear in the environment pane. Typing just the name of the object will print the value of the object in the console pane.

   ```
   a
   b
   ```

3. Because the new objects a and b contain a single numerical value, you can apply calculator commands and true/false questions to the objects. These are objects containing numerical information and not text, so quotation marks are not required and using quotation marks would refer to something different than the objects a and b. It is also important to note here that RStudio is sensitive to capitalization. So A and B are not equal to a and b.
   Type the following commands directly into the console pane:

   ```
   a + b
   a – b
   a * b
   a / b
   a == b
   a != b
   a <= b
   a >= b
   ```

Use objects to build new objects.

1. Build new objects c and d by typing the following command directly into the console pane:

   ```
   c <- a + b
   d <- a * b / c
   ```

   In general you can name an object whatever you like so long as the object name does not begin with a number or a set of reserved words: if, for, next, break, in, etc.

2. Generate a new object with a more descriptive name.
   here.is.an.example.of.an.unnecessarily.long.name <- 1
   To modify an already existing object, use the assignment operator (<-) and the original object's name.

3. The values for a and b are currently 15 and 20 respectively. Imagine that we need to convert these values to percentages. Normally, you would generate a new object such as a.percent and b.percent so as to not lose any data. However, in the example below we replace the old values of a and b with new percentages.
   a <- (15/(15+20)*100)
   b <- (20/(15+20)*100)
   Notice that the object names for a and b don't change in the environment pane, but the values for a and b change after modifying the objects with the commands above.

   One of the remarkable features about using RStudio is its capacity to work with and store multiple objects. We will move to more complicated objects soon, but imagine if your objects a, b, c, and d were all separate spreadsheets. In statistical software like Stata, SPSS, or Excel, it is only possible to work within a single spreadsheet at a time. Because RStudio is object based, it can actually hold multiple spreadsheets so long as each spreadsheet is assigned an object. The number of databases open in RStudio is a only limited by computer memory and what users can keep track of.

F. **Creating and Modifying Vectors in Rstudio**

Usually you won't assign only a single number to an object, as it would be more efficient to just type the number than assign it to an object. Objects in RStudio can actually be many things. In this section of the Lab, we will look at vectors as objects in R. A vector is essentially a list of items. If you were looking at a spreadsheet in Excel, a vector could be a single column or a single row within the spreadsheet, but only one. Vectors are the next step up from a single value, but they are a step below spreadsheets. When you see the word vector, think "ordered list"—a list in which the order of the items could matter.

Perhaps you have to go to the grocery store after work, and you feel compelled to generate your shopping list in RStudio. This might not be the most efficient way to generate a list, but it will certainly introduce the idea of vectors.

The combine command is required to make lists: c()
The combine command tells RStudio that you want to make a vector with all of the items inside of the parentheses of the combine command.

1. Type the following command into the console panel to generate a new object called grocery.list that will be a vector containing four items: apples, milk, eggs, and paper towels. Because apples, milk, eggs, and paper towels are strings and not numbers or objects, we put quotation marks around each string and separate them with commas.

```
grocery.list <- c("apples", "milk", "eggs", "paper towels")
```

2. The object grocery.list now appears in the environment pane, but notice what is different between this object and the other objects. Rather than a single item you see a description of the vector.

3. Now whenever you want to see your grocery list, just type the name of the object directly into the console pane. grocery.list

Here is an operation that you can run on a list of strings.

1. You can alphabetize the list using the sort() command. The command below just prints the alphabetized list in the console pane without actually changing the order of the list.
   ```
   sort(grocery.list)
   ```

   The command below actually changes the order of the original list using the assignment command to replace what is stored in the original object
   grocery.list.

   ```
   grocery.list <- sort(grocery.list)
   ```
   Square brackets [] are useful in vectors. They are used to slice out pieces of the vector, add pieces to the vector, or specify which items in a vector need to be changed.

2. Right now grocery.list contains 4 items. If you wanted to inspect a certain item of the list, then type the number of the item you want in the square brackets. To inspect or retrieve the 1st item of grocery.list type the following command:
   ```
   grocery.list[1]
   ```

   To inspect or retrieve the 3rd item of grocery.list type the following command:
   ```
   grocery.list[3]
   ```

3. Perhaps you forgot some things on your grocery list. You need to add two items: olive oil and soap. The numbers within square brackets will specify where in
   grocery.list you want the new items to go. Since we already have 4 items in
   grocery.list, we want to add items number 5 and number 6.
   ```
   grocery.list[5:6] <- c("olive oil", "soap")
   ```
   Notice that the number of items in this object changes in the environment pane.

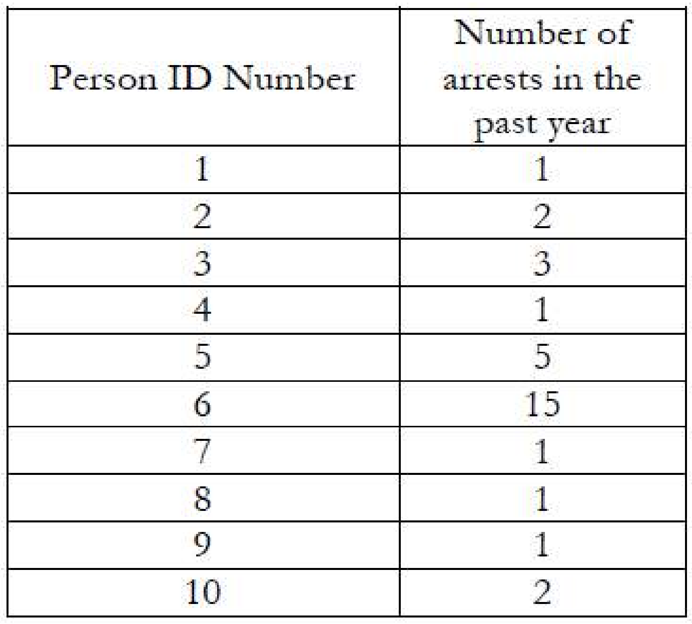
4. You remember that you have a severe allergy to apples, so instead of apples you actually want to buy bananas. The following command will select the 1st item on the list, which is "apples", and replace the word "apples" with the word "bananas".
   ```
   grocery.list[1] <- "bananas"
   ```

5. Remember that you can always inspect your changes by just typing the name of the object directly into the console pane.

```
grocery.list
```

G. Numerical Vectors in Rstudio

Most of the work that you are likely to do with vectors will be with numbers or categorical variables that contain strings but only a few categories of strings (e.g., gender, race). This section of the Lab builds a numerical vector and introduces some functions using that vector.



Person ID Number	Number of arrests in the past year
1	1
2	2
3	3
4	1
5	5
6	15
7	1
8	1
9	1
10	2

In the Lab for the next module you will learn how to load data from existing spreadsheets, but for this example we will need to input the number of arrests in the past year into a vector.

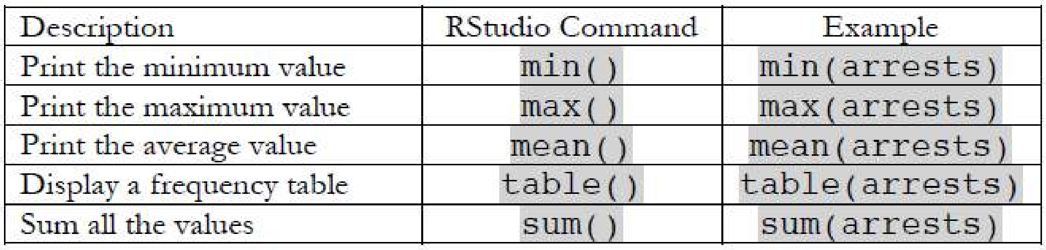
1. Generate a numeric vector based on the table above using the following command.
   ```
   arrests <- c(1, 2, 3, 1, 5, 15, 1, 1, 1, 2)
   ```

2. Uh oh! Person number 6 just got arrested again. Add a 1 to the number of arrests for person #6.
   ```
   arrests[6] <- arrests[6] + 1
   ```

   Notice how the square brackets are used on both the right and the left of the assignment arrow. The right side of the assignment arrow tells RStudio to add 1 to the 6th item in the vector called arrests. The left side of the assignment arrow tells RStudio that this new information should replace the 6th item in the vector called arrests.

3. Try out some of the commands below that were introduced earlier in this lab.
   ```
   arrests + 10
   arrests[1] == arrests[4]
   arrests[c(1,2,3,4,5)] <= arrests[6]
   sort(arrests)
   ```

4. Try out these new commands below that can be used with numerical vectors.



Description	RStudio Command	Example
Print the minimum value	`min()`	`min(arrests)`
Print the maximum value	`max()`	`max(arrests)`
Print the average value	`mean()`	`mean(arrests)`
Display a frequency table	`table()`	`table(arrests)`
Sum all the values	`sum()`	`sum(arrests)`

4. Create a new object that contains the percent of total arrests for each individual.
   arrests.percent <- (arrests/sum(arrests))*100
   table(arrests.percent)

At this point in Module 1: Introduction RStudio, users are hopefully starting to get the feel of working in RStudio, understand some of the logic of how RStudio commands are structured, and how to execute commands in RStudio. In the next part of this Lab, users will learn how to set working directories and execute scripts.

WORKING DIRECTORY

A working directory is the place on a user's computer where files will be retrieved from and sent to. One of the first things required in every RStudio session is to set a working directory. Setting the working directory tells RStudio where to look for files that you want to import and where to export files that you create.

Each Module contains a folder with lab materials. Before setting the working directory, you need to know where the folder is for this Module 1: Introduction where you should have the Lab 1
Files folder.

There are a couple of ways to set the working directory. The first way uses clicking on menus and is numbered in order below:

1. Select the Session menu located toward the middle of the menu bar at the very top of the RStudio program.
2. Select Set Working Directory
3. Select Choose Directory
4. Locate the Lab 1 Files folder on your computer
5. Click open

A second alternative way to set the working directory is to type the entire file path name between the quotation marks of the setwd("") (set working directory) command below. The path name is a statement of where a file or folder is located on your computer. Path names must include the proper slash, colons, drives, etc.

```
setwd("")
```

To check the contents of the working directory of any RStudio session run the directory command below dir(). This will display all of the available files from the current directory in the console pane. If the files listed are incorrect then the working directory has been set incorrectly.

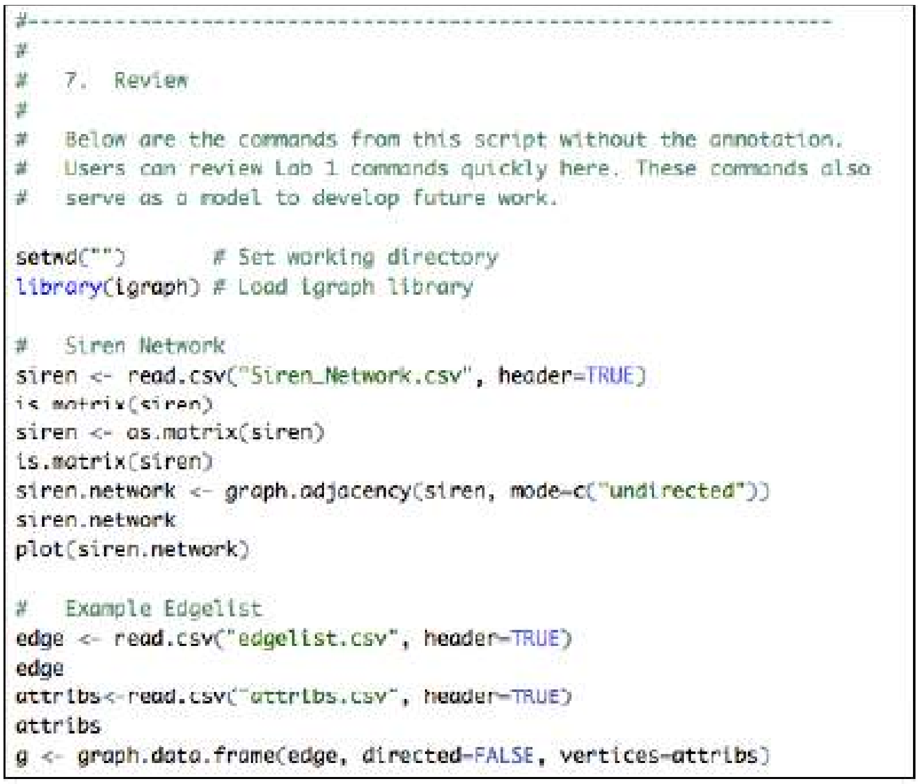
6. Type the following command to list the items in your working directory:
   dir()

You should see the name of one file called, introduction.R. This introduction.R file is called a script, which brings you to the next section of this Lab 1 task.

SCRIPT

Scripts are technically text files in which to write programs to be executed within RStudio. A script is recognizable by its file extension .R. Scripts are similar to do files in Stata or scripts in SPSS. Scripts offer an organized way to save, inspect, and rerun analyses. Scripts can include user written notes that explain or organize commands. Scripts can be opened and inspected in a simple text editing program; however, opening scripts in this way does not permit them to speak with RStudio.

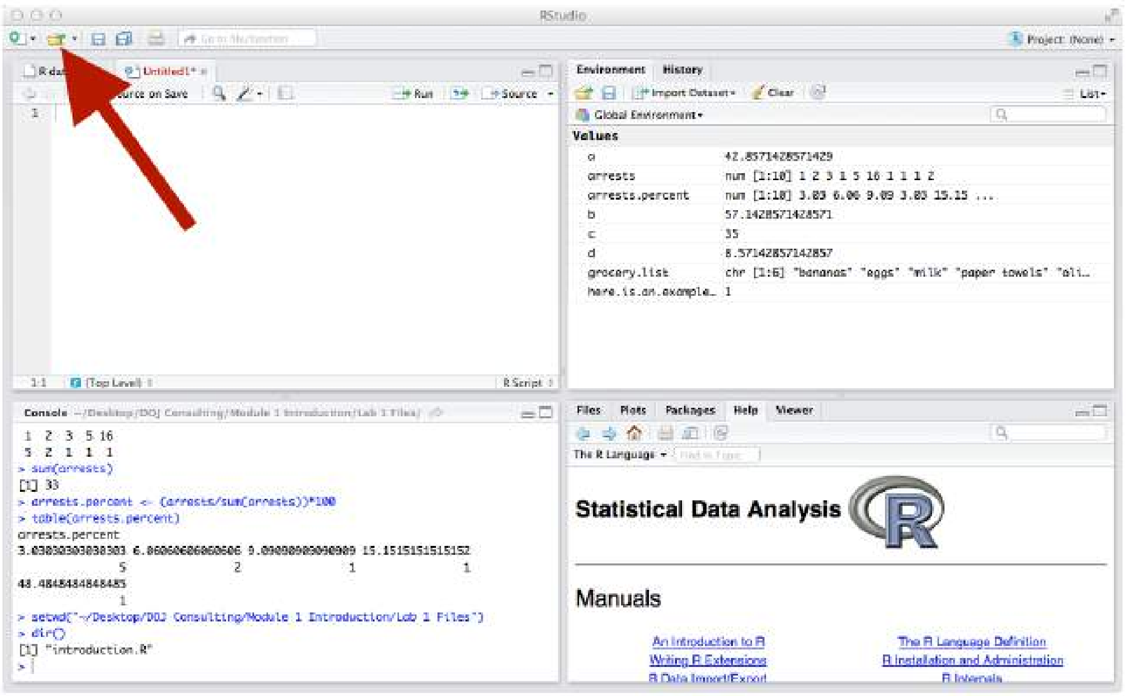
Below is an image showing a section of a script:



```
#-----------------------------------------------------------------
#
#   7.  Review
#
#   Below are the commands from this script without the annotation.
#   Users can review Lab 1 commands quickly here. These commands also
#   serve as a model to develop future work.

setwd("")        # Set working directory
library(igraph) # Load igraph library

#   Siren Network
siren <- read.csv("Siren_Network.csv", header=TRUE)
is.matrix(siren)
siren <- as.matrix(siren)
is.matrix(siren)
siren.network <- graph.adjacency(siren, mode=c("undirected"))
siren.network
plot(siren.network)

#   Example Edgelist
edge <- read.csv("edgelist.csv", header=TRUE)
edge
attribs<-read.csv("attribs.csv", header=TRUE)
attribs
g <- graph.data.frame(edge, directed=FALSE, vertices=attribs)
```

Open the introduction.R script now by following the directions below.

7. In RStudio click on the Open icon shaped like a folder located in the upper-left pane of the RStudio window

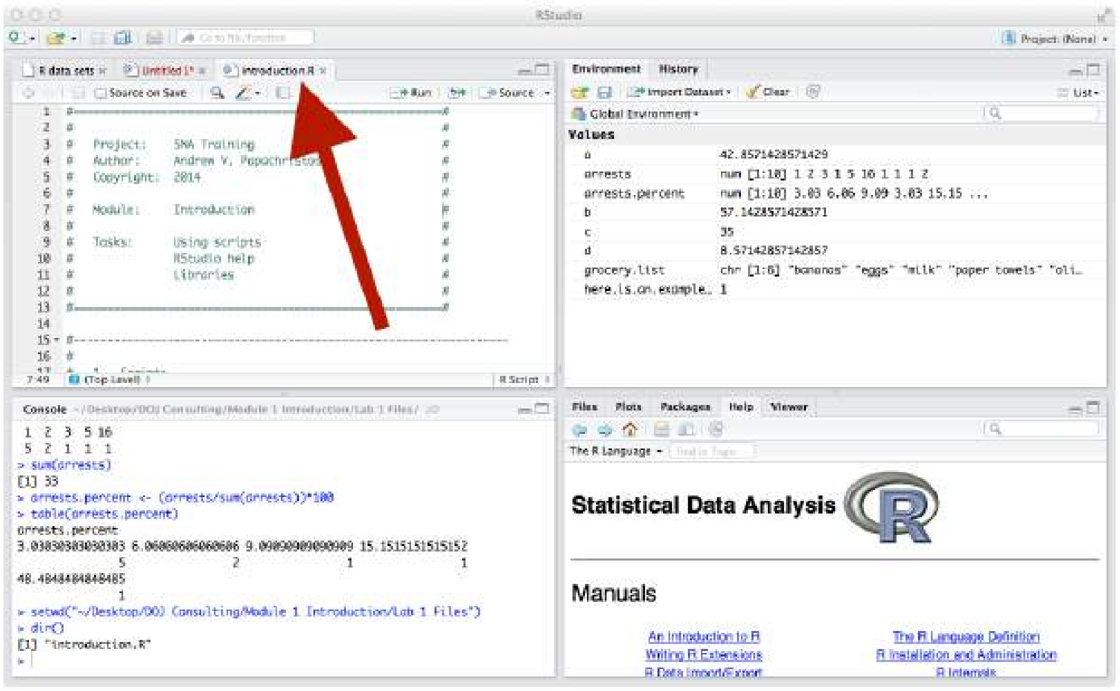


```
Values
  a                      42.8571428571429
  arrests                num [1:10] 1 2 3 1 5 16 1 1 1 2
  arrests.percent        num [1:10] 3.03 6.06 9.09 3.03 15.15 ...
  b                      57.1428571428571
  c                      35
  d                      8.57142857142857
  grocery.list           chr [1:6] "bananas" "eggs" "milk" "paper towels" "oli…
  here.is.an.example…    1
```

```
Console ~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files/
 1  2  3  5 16
 5  2  1  1  1
> sum(arrests)
[1] 33
> arrests.percent <- (arrests/sum(arrests))*100
> table(arrests.percent)
arrests.percent
3.03030303030303 6.06060606060606 9.09090909090909 15.1515151515152
               5                2                1                1
48.4848484848485
               1
> setwd("~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files")
> dir()
[1] "introduction.R"
> 
```

8. Check that the folder is for Lab 1 Files
9. Select the introduction.R script

The introduction.R script should open in the top-left pane. It will look like a text file with the file name at the top.



```
#==============================================================#
#                                                              #
#   Project:     SNA Training                                  #
#   Author:      Andrew V. Papachristos                        #
#   Copyright:   2014                                          #
#                                                              #
#   Module:      Introduction                                  #
#                                                              #
#   Tasks:       Using scripts                                 #
#                RStudio help                                  #
#                Libraries                                     #
#                                                              #
#==============================================================#

#---------------------------------------------------------------------
#
```

```
Values
  a                      42.8571428571429
  arrests                num [1:10] 1 2 3 1 5 16 1 1 1 2
  arrests.percent        num [1:10] 3.03 6.06 9.09 3.03 15.15 ...
  b                      57.1428571428571
  c                      35
  d                      8.57142857142857
  grocery.list           chr [1:6] "bananas" "eggs" "milk" "paper towels" "oli…
  here.is.an.example…    1
```

```
Console ~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files/
 1  2  3  5 16
 5  2  1  1  1
> sum(arrests)
[1] 33
> arrests.percent <- (arrests/sum(arrests))*100
> table(arrests.percent)
arrests.percent
3.03030303030303 6.06060606060606 9.09090909090909 15.1515151515152
               5                2                1                1
48.4848484848485
               1
> setwd("~/Desktop/DOJ Consulting/Module 1 Introduction/Lab 1 Files")
> dir()
[1] "introduction.R"
> 
```

Now that the introduction.R script is open in RStudio, use the introduction.R script to complete the remainder of the Module 1: Introduction RStudio. Instructions are contained within the RStudio script.

Indonesia International Institute for Life Science

[BI2107]

LINUX AND R COMPUTATION

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   - cc. If you have any troubleshoot please contact your supervisor or Building Management
   - dd. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   - ee. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   - ff. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 8

Date : November 5th, 2019
Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 8, students are expected to understand how to run Rmarkdown.
- Markdown is a simple formatting language designed to make authoring content easy for everyone. Rather than write in complex markup code (e.g. HTML or LaTex), you write in plain text with formatting cues.

Procedure

Installation

You can install the R Markdown package from CRAN as follows:

```
install.packages("rmarkdown")

library("rmarkdown")
```

R Markdown files are the source code for rich, reproducible documents. You can transform an R Markdown file in two ways.

1. **knit** - You can *knit* the file. The `rmarkdown` package will call the `knitr` package. `knitr` will run each chunk of R code in the document and append the results of the code to the document next to the code chunk. This workflow saves time and facilitates reproducible reports.

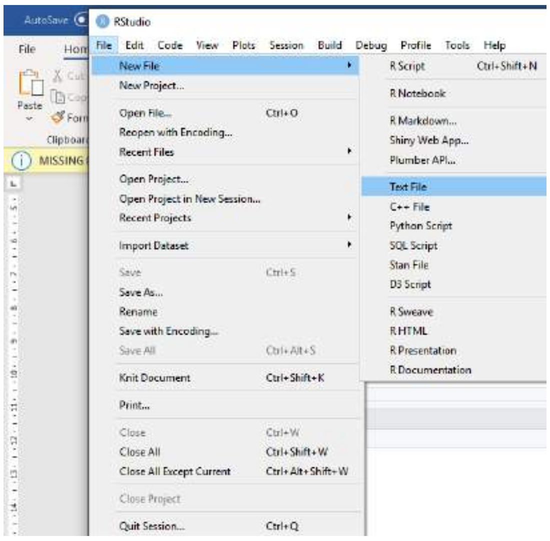
Consider how authors typically include graphs (or tables, or numbers) in a report. The author makes the graph, saves it as a file, and then copy and pastes it into the final report. This process relies on manual labor. If the data changes, the author must repeat the entire process to update the graph.

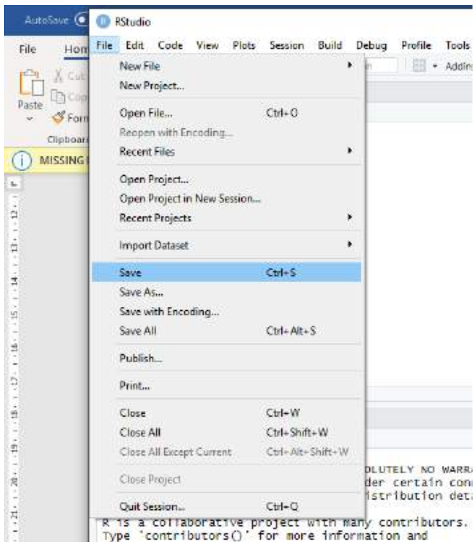
In the R Markdown paradigm, each report contains the code it needs to make its own graphs, tables, numbers, etc. The author can automatically update the report by re-knitting.

2. **convert** - You can *convert* the file. The `rmarkdown` package will use the `pandoc` program to transform the file into a new format. For example, you can convert your .Rmd file into an HTML, PDF, or Microsoft Word file. You can even turn the file into an HTML5 or PDF slideshow. `rmarkdown` will preserve the text, code results, and formatting contained in your original .Rmd file.

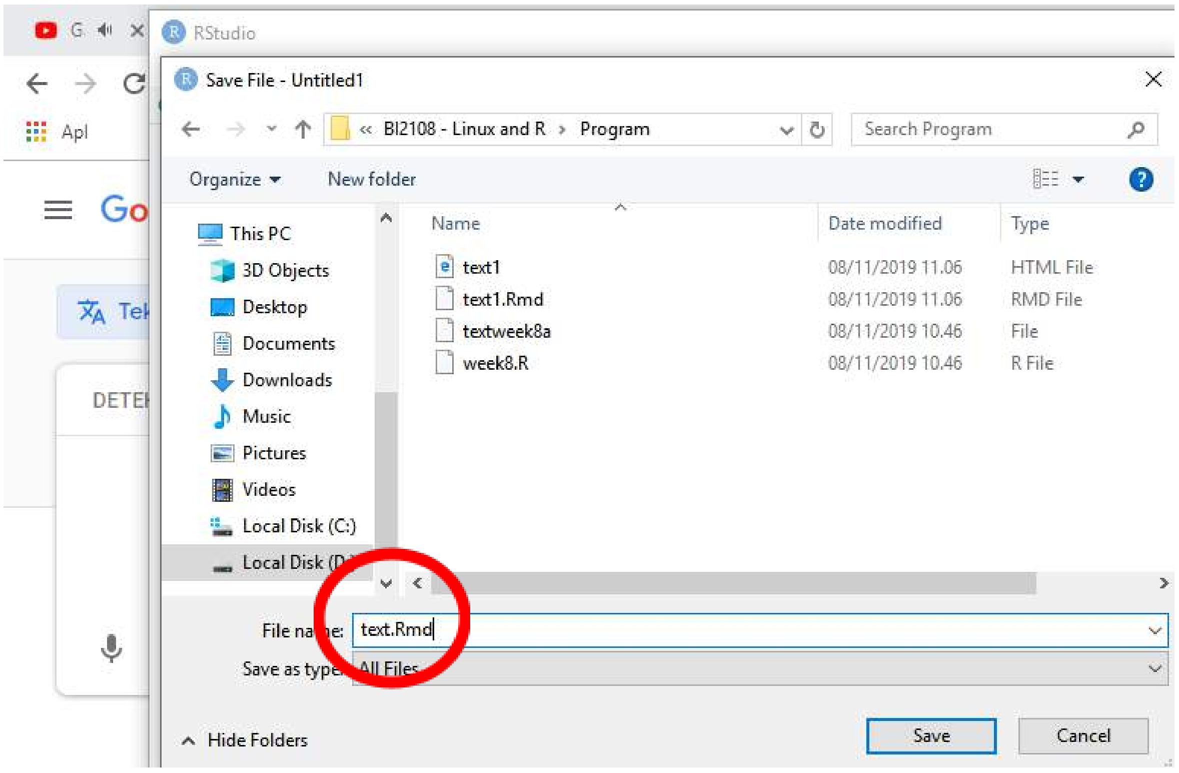
Creating R Markdown File

- To create an R Markdown report, open a plain text file and save it with the extension *.Rmd*. You can open a plain text file in your scripts editor by clicking **File > New File > Text File** in the RStudio toolbar.
- To save the new created file, you can click **File > Save**





- Find a location where you want to save the file then write the file name with extension *.Rmd*



Name	Date modified	Type
text1	08/11/2019 11.06	HTML File
text1.Rmd	08/11/2019 11.06	RMD File
textweek8a	08/11/2019 10.46	File
week8.R	08/11/2019 10.46	R File

File name: text.Rmd
Save as type: All Files

- Then the screen will come up as a R markdown file.

Running R Markdown File

.Rmd files are meant to contain text written in markdown. Markdown is a set of conventions for formatting plain text. You can use markdown to indicate

Emphasis

```
*italic*   **bold**

_italic_   __bold__
```

Headers

```
# Header 1

## Header 2

### Header 3
```

Lists

Unordered List

```
* Item 1
* Item 2
    + Item 2a
    + Item 2b
```

Ordered List

```
1. Item 1
2. Item 2
3. Item 3
```

```
    + Item 3a
    + Item 3b
```

Manual Line Breaks

End a line with two or more spaces:

```
Roses are red,
Violets are blue.
```

Links

Use a plain http address or add a link to a phrase:

```
http://example.com

[linked phrase](http://example.com)
```

Images

Images on the web or local files in the same directory:

```
![alt text](http://example.com/logo.png)

![alt text](figures/img.png)
```

Blockquotes

```
A friend once said:

> It's always better to give
> than to receive.
```

R Code Blocks

R code will be evaluated and printed

~~~
```{r}
summary(cars$dist)
summary(cars$speed)
```
~~~

Inline R Code

```
There were `r nrow(cars)` cars studied
```

Plain Code Blocks

Plain code blocks are displayed in a fixed-width font but not evaulated

~~~
```
This text is displayed verbatim / preformatted
```
~~~

Inline Code

```
We defined the `add` function to
compute the sum of two numbers.
```

LaTeX Equations

Inline Equation

```
$equation$
```

Display Equation

```
$$ equation $$
```

Horizontal Rule / Page Break

Three or more asterisks or dashes:

```
******

------
```

Tables

```
First Header  | Second Header
------------- | -------------
Content Cell  | Content Cell
Content Cell  | Content Cell
```

Reference Style Links and Images

Links

```
A [linked phrase][id].

At the bottom of the document:

[id]: http://example.com/ "Title"
```

Images

```
![alt text][id]

At the bottom of the document:

[id]: figures/img.png "Title"
```

Miscellaneous

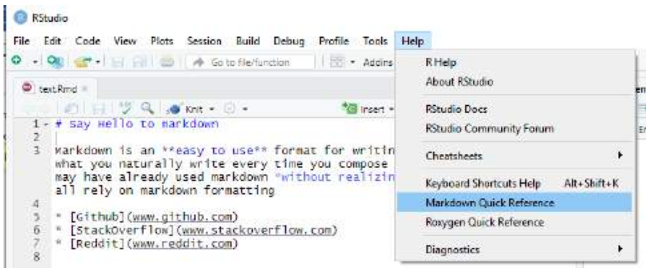
```
superscript^2^

subscript~2~

~~strikethrough~~
```

Typographic Entities

ASCII characters are transformed into typographic HTML entities:

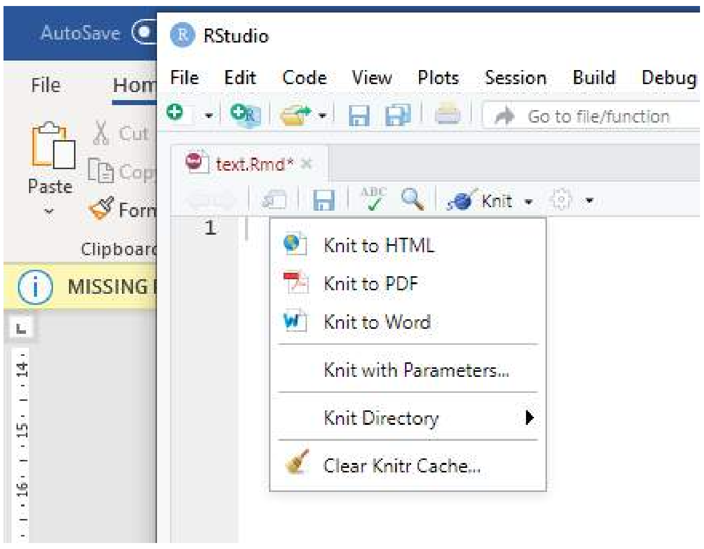
- Straight quotes (" and ') into “curly” quotes
- Dashes (“--” and “---”) into en- and em-dash entities
- Three consecutive dots (“...”) into an ellipsis entity

To access the guide, open a *.md* or *.Rmd* file in RStudio. Then click the question mark that appears at the top of the scripts pane. Next, select “Markdown Quick Reference”. RStudio will open the *Markdown Quick Reference* guide in the Help pane.



```
# say Hello to markdown

Markdown is an **easy to use** format for writin
what you naturally write every time you compose
may have already used markdown *without realizin
all rely on markdown formatting

* [Github](www.github.com)
* [StackOverflow](www.stackoverflow.com)
* [Reddit](www.reddit.com)
```

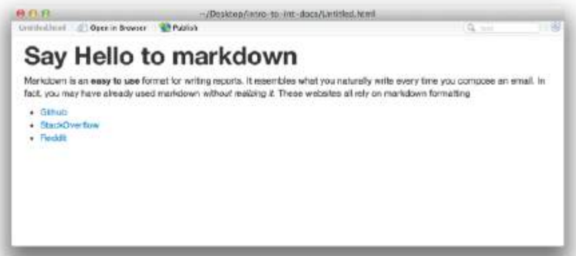
Then, too run the codes, you can click the “Knit” botton by choosing what type of files you want to demonstrate your document.



Example 1:

```
# Say Hello to markdown


Markdown is an **easy to use** format for writing reports. It resembles what
you naturally write every time you compose an email. In fact, you may have
already used markdown *without realizing it*. These websites all rely on
markdown formatting


* [Github](www.github.com)

* [StackOverflow](www.stackoverflow.com)

* [Reddit](www.reddit.com)
```

This how the R markdown wil be showed in html (website) format:


~/Desktop/intro-to-int-docs/Untitled.html

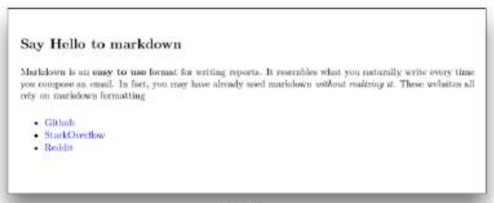
Say Hello to markdown

Markdown is an **easy to use** format for writing reports. It resembles what you naturally write every time you compose an email. In fact, you may have already used markdown *without realizing it*. These websites all rely on markdown formatting

- Github
- StackOverflow
- Reddit

HTML

This how the R markdown wil be showed pdf format:

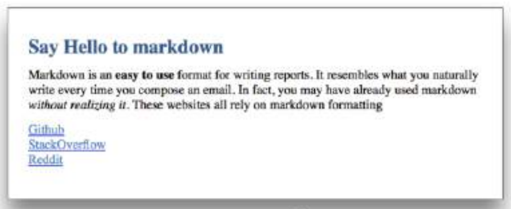

Say Hello to markdown

Markdown is an **easy to use** format for writing reports. It resembles what you naturally write every time you compose an email. In fact, you may have already used markdown *without realizing it*. These websites all rely on markdown formatting

- Github
- StackOverflow
- Reddit

PDF

This how the R markdown wil be showed pdf format:


Say Hello to markdown

Markdown is an **easy to use** format for writing reports. It resembles what you naturally write every time you compose an email. In fact, you may have already used markdown *without realizing it*. These websites all rely on markdown formatting

Github
StackOverflow
Reddit

MS Word

Example 2:

In the **Console**, write and work a mini project.

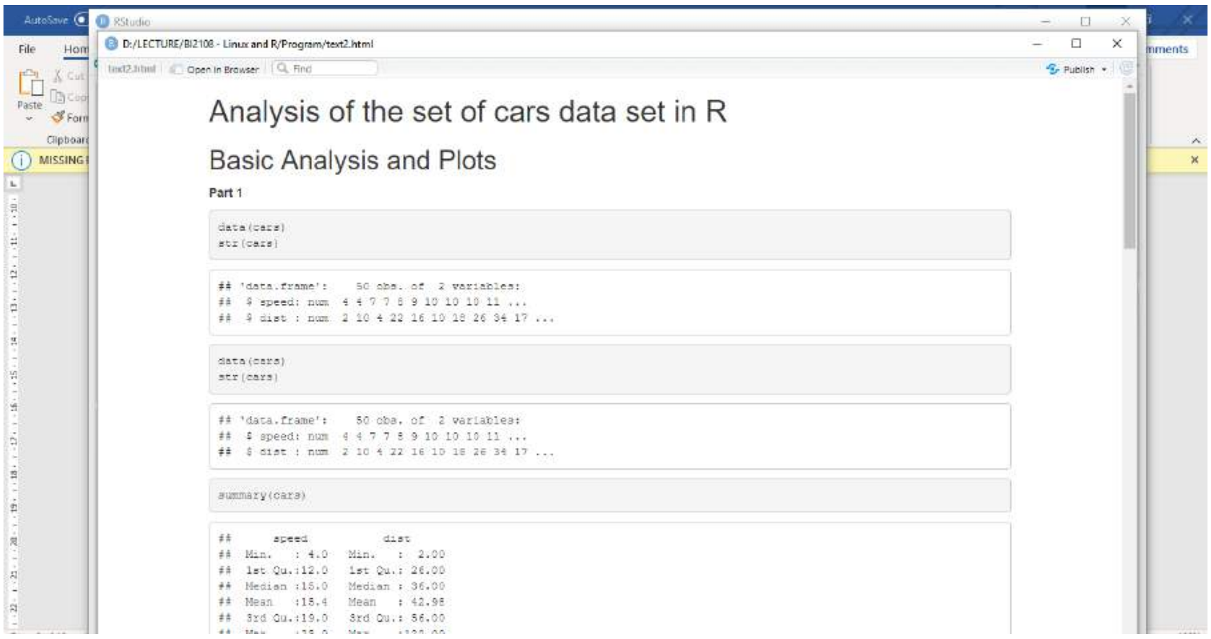
```
data(cars)
str(cars)
summary(cars)
plot(cars)
```

In the **Script**, write codes:

```
# Analysis of the set of cars data set in R
## Basic Analysis and Plots
```

~~~
**Part 1**

```{r}
data(cars)
str(cars)
data(cars)
str(cars)
summary(cars)
plot(cars)
```

**Part 2**

```{r}
hist(cars$speed)
boxplot(cars$dist)
```

The mean speed of cars was ` r mean(cars$speed) `

#The End
~~~

Then run the Knit button. The output will be like the graph below:



Analysis of the set of cars data set in R

Basic Analysis and Plots

Part 1

```
data(cars)
str(cars)
```

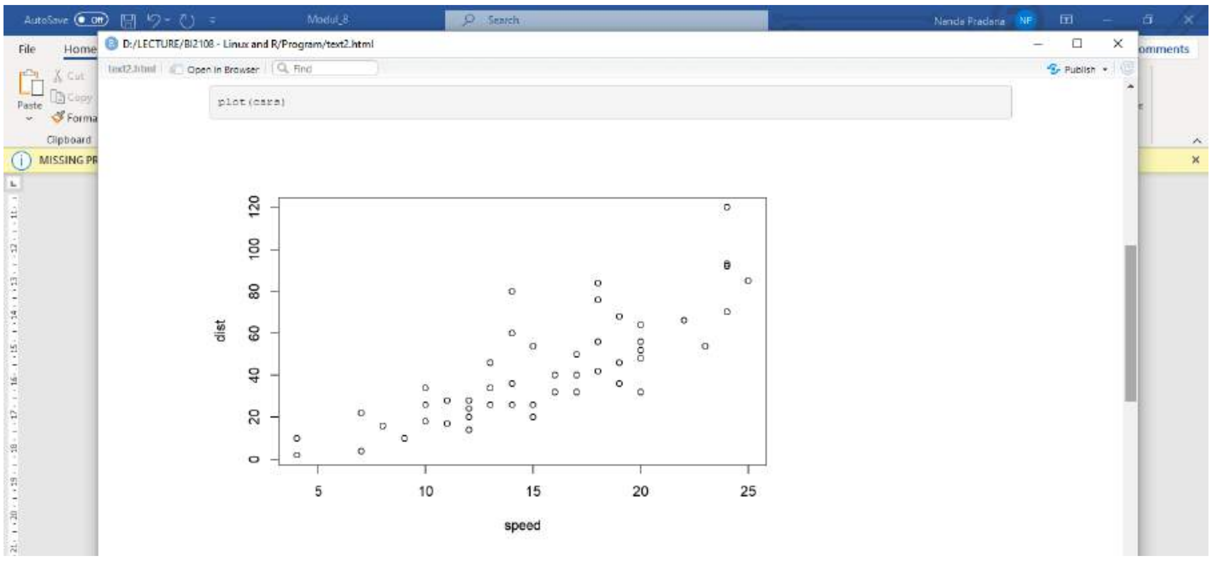
```
## 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

```
data(cars)
str(cars)
```

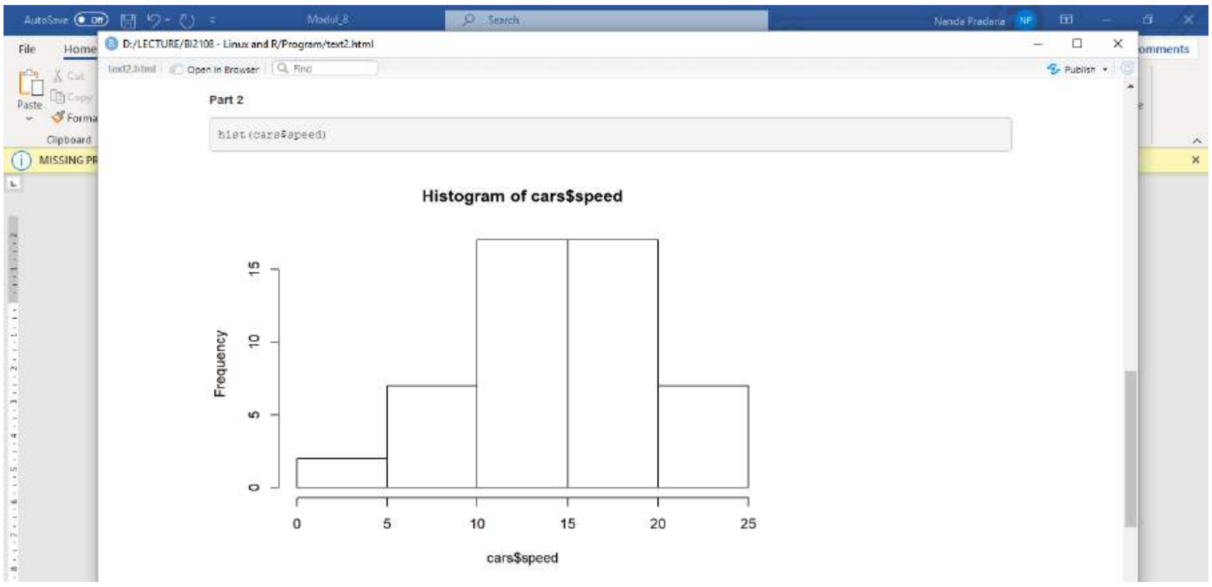
```
## 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
```

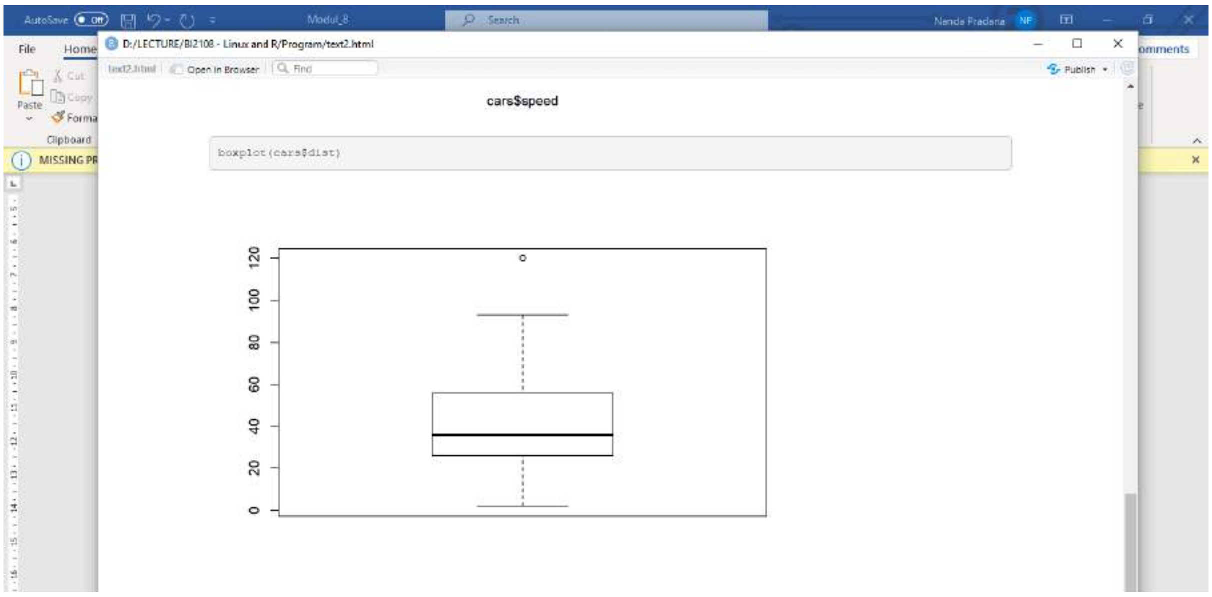


```
plot(cars)
```



Part 2

```
hist(cars$speed)
```



```
boxplot(cars$dist)
```

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   gg. If you have any troubleshoot please contact your supervisor or Building Management
   hh. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   ii. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   jj. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 9

Date : November 12th, 2019

Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 9, students are expected to understand how to run an Application of data manipulation as it is about fitting models to data.

Procedure

First of all, you are recommended to clear all the data in the exist R script's memory before write the new file by typing:

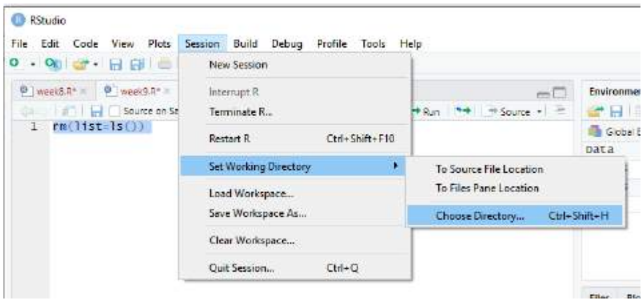
```
rm(list=ls())      # Clear R's memory
```

Then, you are suggested to find the directory where your data will be or has already saved, by typing:

```
setwd('~/DataModule') # Set the working directory
# Replace '~/DataModule' with the name of your own directory
```

The **#** symbol is typed to informed R that the sentences following are not code.

Another step to choose directory can be done using menu in R studio screen.



```
1  rm(list=ls())
```

DATA STRUCTURE

Data structures describe various way of coherently organizing data. The most common data structures in R are:

- vectors
- matrix
- array
- data frame
- time series
- list

VECTOR

```
> x <- c(10, 5, 3, 6)
> x
[1] 10  5  3  6
```

The function c merges an arbitrary number of vectors to one vector. A single number is regarded as a vector of length one.

```
> y <- c(x, 0.55, x, x)
> y
 [1] 10.00  5.00  3.00  6.00  0.55 10.00  5.00  3.00  6.00 10.00  5.00  3.00
[13]  6.00
```

```
> round(y, 3)
 [1] 10.00  5.00  3.00  6.00  0.55 10.00  5.00  3.00  6.00 10.00  5.00  3.00
[13]  6.00
```

The sequence function seq()

The function seq() together with its arguments from, to, by or length is used to generate more general sequences. Specify the beginning and end of the sequence and either specify the length of the sequence or the increment.

```
> u <- seq(from = -3, to = 3, by = 0.5)
> u
 [1] -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0
```

```
> u <- seq(-3, 3, length = 13)
```

The repeat function rep()

The function rep repeats a given vector. The first argument is the vector and the second argument can be a number that indicates how often the vector needs to be repeated.

```
> rep(1:4, 4)
 [1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

```
> rep(1:4, c(2, 2, 2, 2))
[1] 1 1 2 2 3 3 4 4
> rep(1:4, 1:4)
 [1] 1 2 2 3 3 3 4 4 4 4
```

To generate vectors with random elements you can use the functions rnorm or runif. There are more of these functions.

```
> x <- rnorm(10)
> y <- runif(10, 4, 7)
```

MATRICES

Converting vectors to matrices with the dim() function

Use the function dim() to convert a vector into a matrix. It does internally add the special attribute "dim" to the vector.

```
> x <- 1:8
> dim(x) <- c(2, 4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

Generate matrices with the matrix() function

Alternatively use the function matrix() to generate a matrix object.

```
> x <- matrix(1:8, 2, 4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

```
> x <- matrix(1:8, 2, 4, byrow = TRUE)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Binding vectors column and row wise

Use the function cbind() to create a matrix by binding two or more vectors as column vectors.

```
> cbind(c(1, 2, 3), c(4, 5, 6))
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

The function rbind() is used to create a matrix by binding two or more vectors as row vectors.

```
> rbind(c(1, 2, 3), c(4, 5, 6))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

You can multiply a matrix with a vector. The outcome may be surprising:

```
> x <- matrix(1:16, ncol = 4)
> y <- 7:10
> x * y
     [,1] [,2] [,3] [,4]
[1,]    7   35   63   91
[2,]   16   48   80  112
[3,]   27   63   99  135
[4,]   40   80  120  160
```

```
> x <- matrix(1:28, ncol = 4)
> y <- 7:10
> x * y
     [,1] [,2] [,3] [,4]
[1,]    7   80  135  176
[2,]   16   63  160  207
[3,]   27   80  119  240
[4,]   40   99  144  175
[5,]   35  120  171  208
[6,]   48   91  200  243
[7,]   63  112  147  280
```

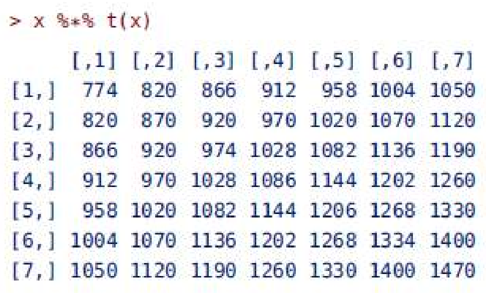
Matrix multiplication

To perform a matrix multiplication in the mathematical sense, use the operator: %*%. The dimensions of the two matrices must conform. In the following example the dimensions are wrong:

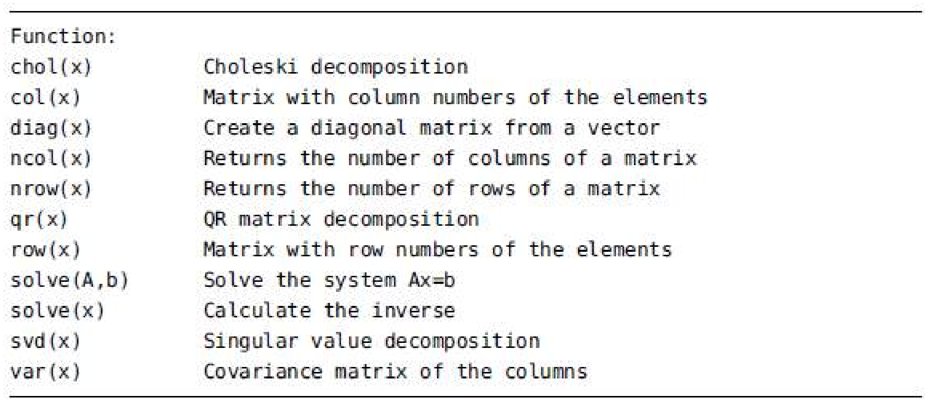
```
x <- matrix(1:8, ncol = 2)
x %*% x
Error in x %*% x : non-conformable arguments
```

The transposed matrix

A matrix multiplied with its transposed t(x) always works.



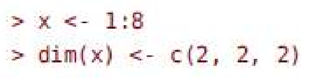
```
> x %*% t(x)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]  774  820  866  912  958 1004 1050
[2,]  820  870  920  970 1020 1070 1120
[3,]  866  920  974 1028 1082 1136 1190
[4,]  912  970 1028 1086 1144 1202 1260
[5,]  958 1020 1082 1144 1206 1268 1330
[6,] 1004 1070 1136 1202 1268 1334 1400
[7,] 1050 1120 1190 1260 1330 1400 1470
```



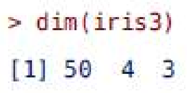
Function:	
chol(x)	Choleski decomposition
col(x)	Matrix with column numbers of the elements
diag(x)	Create a diagonal matrix from a vector
ncol(x)	Returns the number of columns of a matrix
nrow(x)	Returns the number of rows of a matrix
qr(x)	QR matrix decomposition
row(x)	Matrix with row numbers of the elements
solve(A,b)	Solve the system Ax=b
solve(x)	Calculate the inverse
svd(x)	Singular value decomposition
var(x)	Covariance matrix of the columns

ARRAY

Arrays are vectors with a dimension attribute specifying more than two dimensions. A vector is a one-dimensional array and a matrix is a two dimensional array. As with vectors and matrices, all the elements of an array must be of the same data type. An example of an array is the threedimensional array 'iris3', which is a built-in data object in R. A three dimensional array can be regarded as a block of numbers.



```
> x <- 1:8
> dim(x) <- c(2, 2, 2)
```



```
> dim(iris3)
[1] 50  4  3
```

The function array() is used to create an array object

```
> newarray <- array(c(1:8, 11:18, 111:118), dim = c(2, 4, 3))
> newarray
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

, , 2

     [,1] [,2] [,3] [,4]
[1,]   11   13   15   17
[2,]   12   14   16   18

, , 3

     [,1] [,2] [,3] [,4]
[1,]  111  113  115  117
[2,]  112  114  116  118
```

DATA FRAME

Data frames can be regarded as lists with element of the same length that are represented in a two dimensional object. Data frames can have columns of different data types and are the most convenient data structure for data analysis in R. In fact, most statistical modeling routines in R require a data frame as input.

```
> data(longley)
> longley


> rownames(longley)

 [1] "1947" "1948" "1949" "1950" "1951" "1952" "1953" "1954" "1955" "1956"
[11] "1957" "1958" "1959" "1960" "1961" "1962"

> names(longley)

[1] "GNP.deflator" "GNP"          "Unemployed"   "Armed.Forces" "Population"
[6] "Year"         "Employed"
```

DATA MANIPULATION

Sorting and ordering vectors

Sorting a vector: To sort a vector in increasing order, use the function sort(). You can also use this function to sort in decreasing order by using the argument decrease = TRUE.

```
> x1 <- c(2, 6, 4, 5, 5, 8, 8, 1, 3, 0)
> length(x1)
[1] 10

> x2 <- sort(x1)
> x3 <- sort(x1, decreasing = TRUE)
```

*Ordering a vector:*With the function order you can produce a permutation vector which indicates how to sort the input vector in ascending order. If you have two vectors x and y, you can sort x and permute y in such a way that the elements have the same order as the sorted vector x.

```
> x <- rnorm(10)
> y <- 1:10
> z <- order(x)

> sort(x)
 [1] -2.03190 -1.66859 -1.54659 -0.81197 -0.66099 -0.50070 -0.30437 -0.17851
 [9]  0.27800  0.35339
```

Change the order of elements of y

```
> y[z]
 [1]  9 10  5  7  1  4  8  3  2  6
```

Reversing a vector: The function rev reverses the order of vector elements. rev(sort(x)) is a sorted vector in descending order.

```
> x <- round(rnorm(10), 2)
> rev(sort(x))
 [1]  1.12  1.02  1.01  0.66  0.59  0.52 -0.42 -0.53 -0.73 -2.02
```

Making vectors unique

The function unique returns a vector which only contains the unique values of the input vector. The function duplicated() returns TRUE or FALSE for every element depending on whether or not that element has previously appeared in the vector.

```
> x <- c(2, 6, 4, 5, 5, 8, 8, 1, 3, 0)
> unique(x)
[1] 2 6 4 5 8 1 3 0


> duplicated(x)
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
```

Differencing a vector

Our last example of a vector manipulation function is the function diff. This returns a vector which contains the differences between the consecutive input elements.

```
> x <- c(1, 3, 5, 8, 15)
> diff(x)

[1] 2 2 3 7
```

```
> x <- c(1, 3, 5, 8, 15)
> diff(x, lag = 2)

[1]  4  5 10
```

Subsetting with two columns
We can also subset a matrix X with two columns. A row of X consists of two numbers, each row of X selects a matrix element of X. The result is a vector with the selected elements from X.

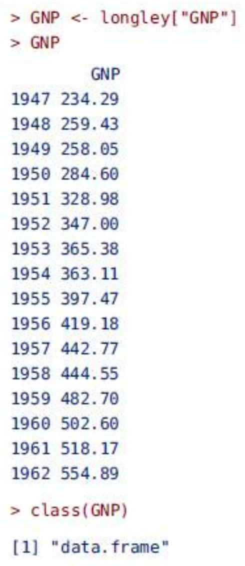
```
> X <- matrix(1:36, ncol = 6)
> X
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    7   13   19   25   31
[2,]    2    8   14   20   26   32
[3,]    3    9   15   21   27   33
[4,]    4   10   16   22   28   34
[5,]    5   11   17   23   29   35
[6,]    6   12   18   24   30   36
> INDEX <- cbind(c(1, 2, 5), c(3, 4, 4))
> INDEX
     [,1] [,2]
[1,]    1    3
[2,]    2    4
[3,]    5    4
> X[INDEX]
[1] 13 20 23
```

Subsetting by a single number or a vector of numbers
What happens when we subset a matrix by a single number or one vector of numbers? In this case the matrix is treated as a vector where all the columns are stacked.

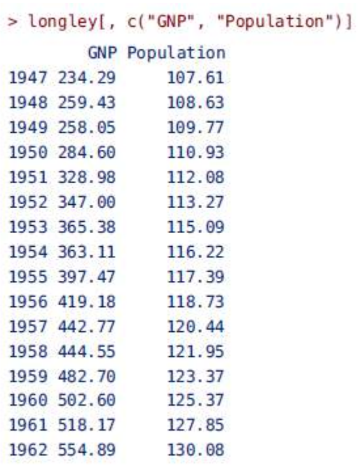
```
> X <- matrix(1:36, ncol = 6)
> X[3]
[1] 3
> X[9]
[1] 9
> X[36]
[1] 36
> X[21:30]
 [1] 21 22 23 24 25 26 27 28 29 30
```

Extracting data from data frames
To select a specific column from a data frame use the $ symbol or double square brackets and quotes:

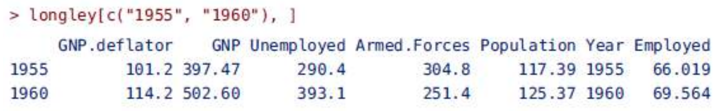
```
> GNP <- longley$GNP
> GNP
 [1] 234.29 259.43 258.05 284.60 328.98 347.00 365.38 363.11 397.47 419.18
[11] 442.77 444.55 482.70 502.60 518.17 554.89
> GNP <- longley[["GNP"]]
> GNP
 [1] 234.29 259.43 258.05 284.60 328.98 347.00 365.38 363.11 397.47 419.18
[11] 442.77 444.55 482.70 502.60 518.17 554.89
> class(GNP)
[1] "numeric"
```



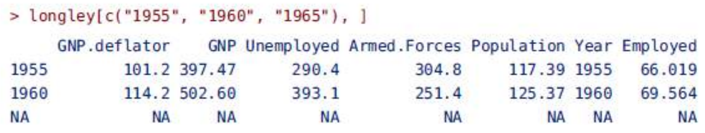
```
> GNP <- longley["GNP"]
> GNP
        GNP
1947 234.29
1948 259.43
1949 258.05
1950 284.60
1951 328.98
1952 347.00
1953 365.38
1954 363.11
1955 397.47
1956 419.18
1957 442.77
1958 444.55
1959 482.70
1960 502.60
1961 518.17
1962 554.89
> class(GNP)
[1] "data.frame"
```



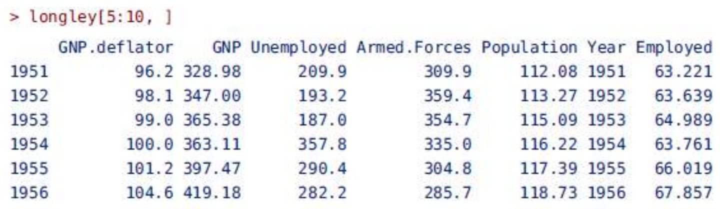
```
> longley[, c("GNP", "Population")]
        GNP Population
1947 234.29     107.61
1948 259.43     108.63
1949 258.05     109.77
1950 284.60     110.93
1951 328.98     112.08
1952 347.00     113.27
1953 365.38     115.09
1954 363.11     116.22
1955 397.47     117.39
1956 419.18     118.73
1957 442.77     120.44
1958 444.55     121.95
1959 482.70     123.37
1960 502.60     125.37
1961 518.17     127.85
1962 554.89     130.08
```



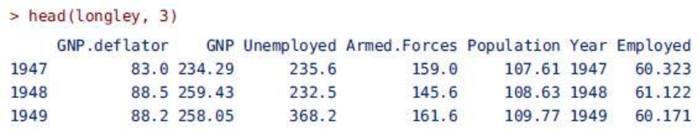
```
> longley[c("1955", "1960"), ]
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1955        101.2 397.47      290.4        304.8     117.39 1955   66.019
1960        114.2 502.60      393.1        251.4     125.37 1960   69.564
```



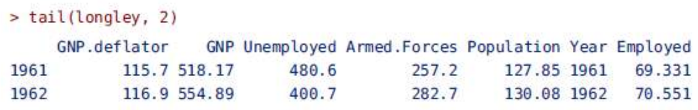
```
> longley[c("1955", "1960", "1965"), ]
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1955        101.2 397.47      290.4        304.8     117.39 1955   66.019
1960        114.2 502.60      393.1        251.4     125.37 1960   69.564
NA             NA     NA         NA           NA         NA   NA       NA
```



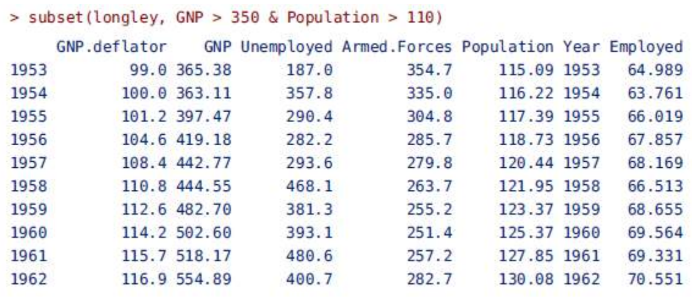
```
> longley[5:10, ]
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1951         96.2 328.98      209.9        309.9     112.08 1951   63.221
1952         98.1 347.00      193.2        359.4     113.27 1952   63.639
1953         99.0 365.38      187.0        354.7     115.09 1953   64.989
1954        100.0 363.11      357.8        335.0     116.22 1954   63.761
1955        101.2 397.47      290.4        304.8     117.39 1955   66.019
1956        104.6 419.18      282.2        285.7     118.73 1956   67.857
```



```
> head(longley, 3)
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1947         83.0 234.29      235.6        159.0     107.61 1947   60.323
1948         88.5 259.43      232.5        145.6     108.63 1948   61.122
1949         88.2 258.05      368.2        161.6     109.77 1949   60.171
```



```
> tail(longley, 2)
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1961        115.7 518.17      480.6        257.2     127.85 1961   69.331
1962        116.9 554.89      400.7        282.7     130.08 1962   70.551
```



```
> subset(longley, GNP > 350 & Population > 110)
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1953         99.0 365.38      187.0        354.7     115.09 1953   64.989
1954        100.0 363.11      357.8        335.0     116.22 1954   63.761
1955        101.2 397.47      290.4        304.8     117.39 1955   66.019
1956        104.6 419.18      282.2        285.7     118.73 1956   67.857
1957        108.4 442.77      293.6        279.8     120.44 1957   68.169
1958        110.8 444.55      468.1        263.7     121.95 1958   66.513
1959        112.6 482.70      381.3        255.2     123.37 1959   68.655
1960        114.2 502.60      393.1        251.4     125.37 1960   69.564
1961        115.7 518.17      480.6        257.2     127.85 1961   69.331
1962        116.9 554.89      400.7        282.7     130.08 1962   70.551
```

Adding columns to a data frame

The function cbind can be used to add additional columns to a data frame. For example, the ratio of the GNP and the population

```
> gnpPop <- round(longley[, "GNP"]/longley[, "Population"], 2)
> longley <- cbind(longley, GNP.POP = gnpPop)
> longley
     GNP.deflator    GNP Unemployed Armed.Forces Population Year Employed
1947         83.0 234.29      235.6        159.0     107.61 1947   60.323
1948         88.5 259.43      232.5        145.6     108.63 1948   61.122
1949         88.2 258.05      368.2        161.6     109.77 1949   60.171
1950         89.5 284.60      335.1        165.0     110.93 1950   61.187
1951         96.2 328.98      209.9        309.9     112.08 1951   63.221
1952         98.1 347.00      193.2        359.4     113.27 1952   63.639
1953         99.0 365.38      187.0        354.7     115.09 1953   64.989
1954        100.0 363.11      357.8        335.0     116.22 1954   63.761
1955        101.2 397.47      290.4        304.8     117.39 1955   66.019
```

Merging data frames

Two data frames can be merged into one data frame using the function merge. 3 If the original data frames contain identical columns, these columns only appear once in the merged data frame. Consider the following two data frames:

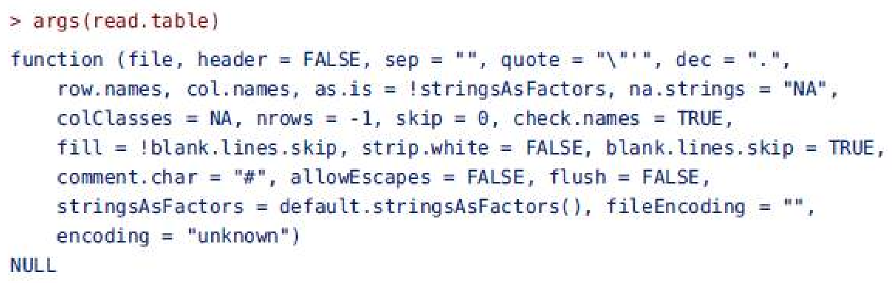
```
> long1 <- longley[1:6, c("Year", "Population", "Armed.Forces")]
> long1
     Year Population Armed.Forces
1947 1947     107.61        159.0
1948 1948     108.63        145.6
1949 1949     109.77        161.6
1950 1950     110.93        165.0
1951 1951     112.08        309.9
1952 1952     113.27        359.4
```

```
> long2 <- longley[1:6, c("Year", "GNP", "Unemployed")]
> long2
     Year    GNP Unemployed
1947 1947 234.29      235.6
1948 1948 259.43      232.5
1949 1949 258.05      368.2
1950 1950 284.60      335.1
1951 1951 328.98      209.9
1952 1952 347.00      193.2
> long3 <- merge(long1, long2)
> long3
  Year Population Armed.Forces    GNP Unemployed
1 1947     107.61        159.0 234.29      235.6
2 1948     108.63        145.6 259.43      232.5
3 1949     109.77        161.6 258.05      368.2
4 1950     110.93        165.0 284.60      335.1
5 1951     112.08        309.9 328.98      209.9
6 1952     113.27        359.4 347.00      193.2
```

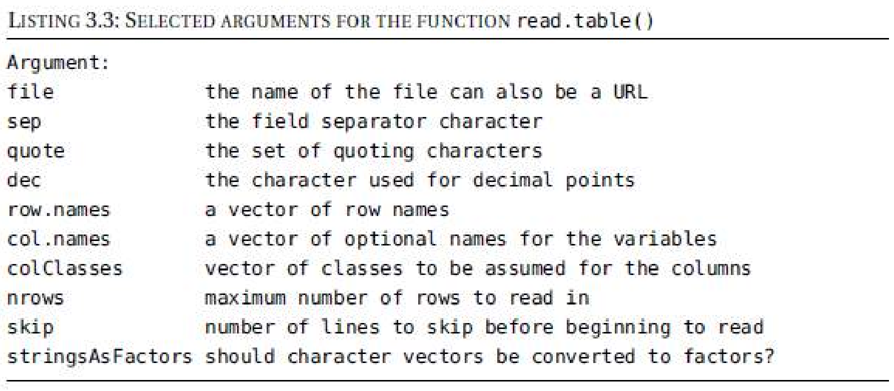
IMPORTING DATA

READING FROM A TEXT FILE WITH read.table()

If we want to import just the data part as a data frame,we can use the function read.table() and skip the header lines. The function has a whole bundle of arguments, e.g. to specify the header, the column separator, the number of lines to skip, the data types of the columns, etc.

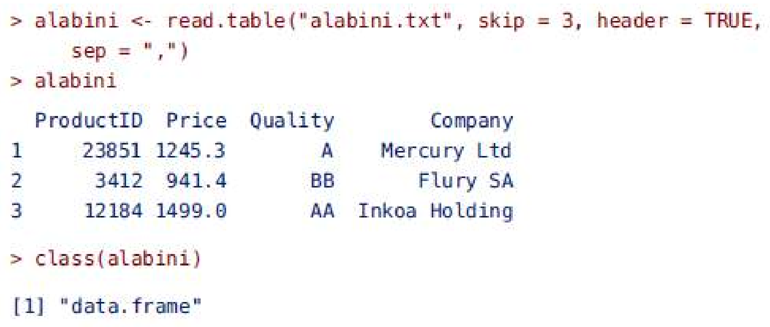


```
> args(read.table)

function (file, header = FALSE, sep = "", quote = "\"'", dec = ".",
    row.names, col.names, as.is = !stringsAsFactors, na.strings = "NA",
    colClasses = NA, nrows = -1, skip = 0, check.names = TRUE,
    fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE,
    comment.char = "#", allowEscapes = FALSE, flush = FALSE,
    stringsAsFactors = default.stringsAsFactors(), fileEncoding = "",
    encoding = "unknown")
NULL
```



LISTING 3.3: SELECTED ARGUMENTS FOR THE FUNCTION read.table()

Argument:	
file	the name of the file can also be a URL
sep	the field separator character
quote	the set of quoting characters
dec	the character used for decimal points
row.names	a vector of row names
col.names	a vector of optional names for the variables
colClasses	vector of classes to be assumed for the columns
nrows	maximum number of rows to read in
skip	number of lines to skip before beginning to read
stringsAsFactors	should character vectors be converted to factors?



```
> alabini <- read.table("alabini.txt", skip = 3, header = TRUE,
     sep = ",")
> alabini

  ProductID  Price  Quality       Company
1     23851 1245.3        A    Mercury Ltd
2      3412  941.4       BB       Flury SA
3     12184 1499.0       AA  Inkoa Holding

> class(alabini)

[1] "data.frame"
```

IMPORTING EXAMPLE DATA FILES

The function data() loads specified data sets, or lists the available data sets. Four formats of data files are supported:

Argument:	
.R, .r	these files are read with source() with the working directory changed temporarily to the directory containing the respective file
.RData, .rd	these files are read with the function load()
.tab, .txt, .TXT	these files are read with the function read.table(..., header = TRUE)}, and hence result in a data frame.
.csv, .CSV	these files are read with the function read.table(..., header = TRUE, sep = ";") also result in a data frame.

DATA MANIPULATION WITH DPLYR

Overview

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- mutate() adds new variables that are functions of existing variables
- select() picks variables based on their names.
- filter() picks cases based on their values.
- summarise() reduces multiple values down to a single summary.
- arrange() changes the ordering of the rows.

These all combine naturally with group_by() which allows you to perform any operation "by group". You can learn more about them in vignette("dplyr"). As well as these single-table verbs, dplyr also provides a variety of two-table verbs, which you can learn about in vignette("two-table").

dplyr is designed to abstract over how the data is stored. That means as well as working with local data frames, you can also work with remote database tables, using exactly the same R code. Install the dbplyr package then read vignette("databases", package = "dbplyr").

If you are new to dplyr, the best place to start is the data import chapter in R for data science.

Installation

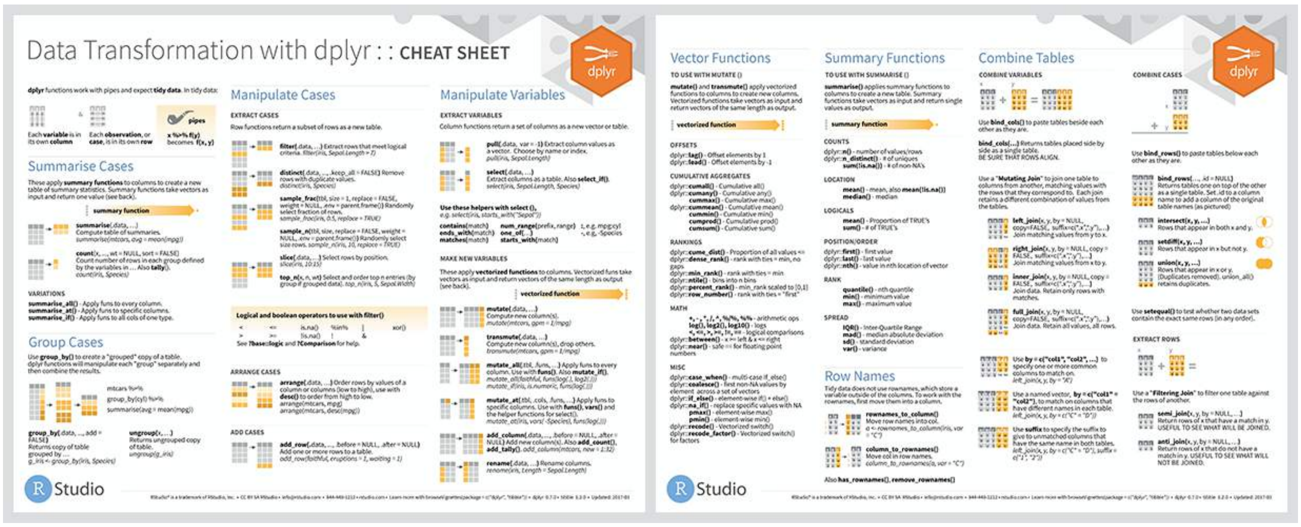
```
# The easiest way to get dplyr is to install the whole tidyverse:
install.packages("tidyverse")

# Alternatively, install just dplyr:
install.packages("dplyr")
```

Development version

To get a bug fix, or use a feature from the development version, you can install dplyr from GitHub.

```
# install.packages("devtools")
devtools::install_github("tidyverse/dplyr")
```

Cheatsheet



Data Transformation with dplyr : : CHEAT SHEET

Manipulate Cases

Manipulate Variables

Summarise Cases

Group Cases

Vector Functions

Summary Functions

Combine Tables

Row Names

Usage

```
library(dplyr)
```

```
starwars %>%
  filter(species == "Droid")
#> # A tibble: 5 x 13
#>   name  height  mass hair_color skin_color eye_color birth_year gender
#>   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
#> 1 C-3PO    167    75 <NA>       gold       yellow           112 <NA>
#> 2 R2-D2     96    32 <NA>       white, bl… red               33 <NA>
#> 3 R5-D4     97    32 <NA>       white, red red               NA <NA>
#> 4 IG-88    200   140 none       metal      red               15 none
#> 5 BB8       NA    NA none       none       black             NA none
#> # … with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#> #   vehicles <list>, starships <list>

starwars %>%
  select(name, ends_with("color"))
#> # A tibble: 87 x 4
#>   name           hair_color skin_color  eye_color
#>   <chr>          <chr>      <chr>       <chr>
#> 1 Luke Skywalker blond      fair        blue
#> 2 C-3PO          <NA>       gold        yellow
#> 3 R2-D2          <NA>       white, blue red
#> 4 Darth Vader    none       white       yellow
#> 5 Leia Organa    brown      light       brown
#> # … with 82 more rows

starwars %>%
  mutate(name, bmi = mass / ((height / 100)  ^ 2)) %>%
  select(name:mass, bmi)
#> # A tibble: 87 x 4
#>   name           height  mass   bmi
#>   <chr>           <int> <dbl> <dbl>
#> 1 Luke Skywalker    172    77  26.0
#> 2 C-3PO             167    75  26.9
#> 3 R2-D2              96    32  34.7
#> 4 Darth Vader       202   136  33.3
#> 5 Leia Organa       150    49  21.8
#> # … with 82 more rows

starwars %>%
  arrange(desc(mass))
#> # A tibble: 87 x 13
#>   name  height  mass hair_color skin_color eye_color birth_year gender
#>   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
#> 1 Jabb…    175  1358 <NA>       green-tan… orange           600 herma…
#> 2 Grie…    216   159 none       brown, wh… green, y…         NA male
#> 3 IG-88    200   140 none       metal      red            15   none
#> 4 Dart…    202   136 none       white      yellow         41.9 male
#> 5 Tarf…    234   136 brown      brown      blue           NA   male
#> # … with 82 more rows, and 5 more variables: homeworld <chr>,
#> #   species <chr>, films <list>, vehicles <list>, starships <list>

starwars %>%
  group_by(species) %>%
  summarise(
    n = n(),
    mass = mean(mass, na.rm = TRUE)
  ) %>%
  filter(n > 1,
         mass > 50)
#> # A tibble: 8 x 3
#>   species      n  mass
#>   <chr>    <int> <dbl>
#> 1 Droid        5  69.8
#> 2 Gungan       3  74
#> 3 Human       35  82.8
#> 4 Kaminoan     2  88
#> 5 Mirialan     2  53.1
#> # … with 3 more rows
```

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   kk. If you have any troubleshoot please contact your supervisor or Building Management
   ll. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   mm. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   nn. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 10

Date : November 12th, 2019
Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 9, students are expected to understand how to run an Application of data manipulation as it is about fitting models to data.

Procedure

Plotting

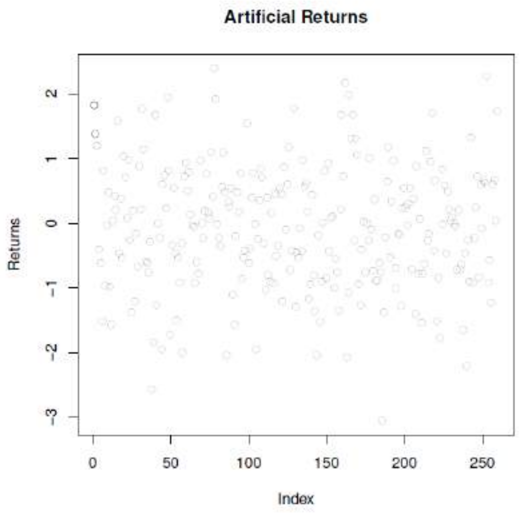
The most elementary plot function is plot. In its simplest formit creates a scatterplot of two input vectors. Let us consider an integer index ranging from 1 to 260 and an artificial log-return series of the same length simulated from normal random variates. Note that the cumulative values form a random walk process.

```
> set.seed(4711)
> Index <- 1:260
> Returns <- rnorm(260)
> plot(x = Index, y = Returns)
```

set.seed()	: is a code used to make data generated having constant orders.
rnorm()	: is used to generate data with normal distribution.
plot(x, y)	: is used to create a plot for two variable data.

```
> plot(x = Index, y = Returns, main = "Artificial Returns")
```

plot(x, y, main =””)   : is to create a plot with a title use the argument main in the plot function.

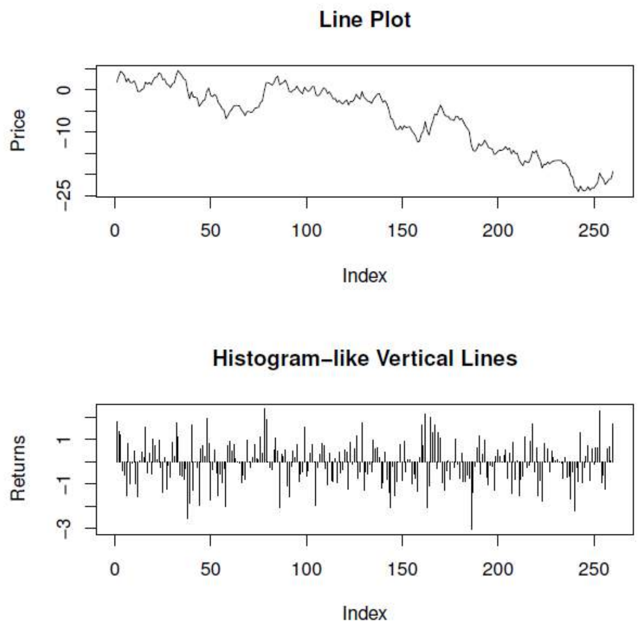


A simple line plot

For a simple line plot which connects consecutive pointswe have to set the option type = "l" in the plot() function. Let us return to the previous example and let us replot the data points in a different way.

```
> par(mfrow = c(2, 1))
> Price = cumsum(Returns)
> plot(Index, Price, type = "l", main = "Line Plot")
> plot(Index, Returns, type = "h", main = "Histogram-like Vertical Lines")
```

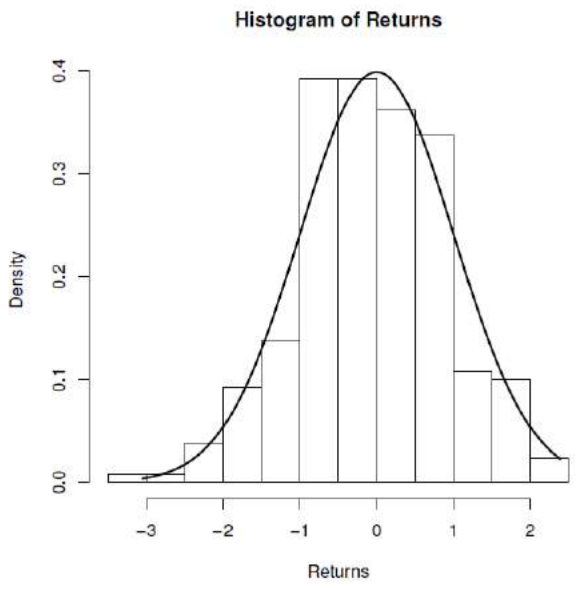
par(mfrow = c(,))   : is a code to create more than one figure in the screen



Drawing functions or expressions

In case of drawing functions or expressions, the function curve can be handy, it takes some work away to produce a line graph. We show this displaying the density of the log-returns underlying a histogram plot.

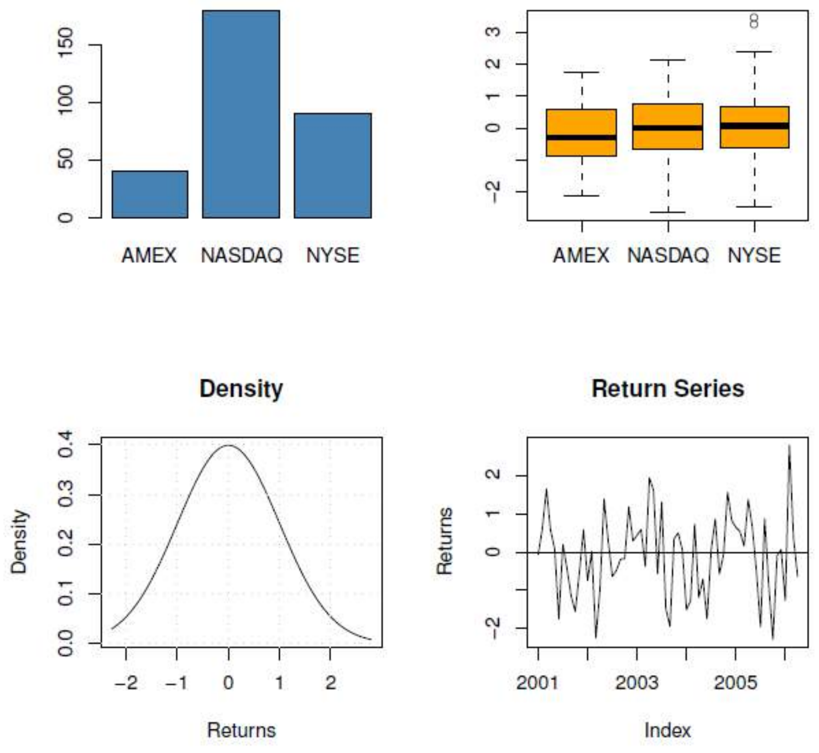
```
> hist(Returns, probability = TRUE)
> curve(dnorm(x), min(Returns), max(Returns), add = TRUE, lwd = 2)
```



```
Function:
plot(xf)              creates a bar plot if xf is a vector of data type
                      factor
plot(xf, y)           creates box-and-whisker plots of the numeric data
                      in y for each level of xf
plot(x.df)            all columns of the data frame x.df are plotted
                      against each other
plot(ts)              creates a time series plot if ts is a time series
                      object


plot(xdate, yval)     if xdate is a Date object R will plot yval with a
                      suitable x-axis
plot(xpos, y)         creates a scatterplot; xpos is a POSIXct object
                      and y a numeric vector
plot(f, low, up)      creates a graph of the function f between low and
                      up
```

The code below shows four examples of the different uses of the function `plot()`.

```
> factorData <- factor(sample(c(rep("AMEX", times = 40), rep("NASDAQ",
     times = 180), rep("NYSE", times = 90))))

> tsData = ts(matrix(rnorm(64), 64, 1), start = c(2001, 1), frequency = 12)

> plot(factorData, col = "steelblue")
> plot(factorData, rnorm(length(factorData)), col = "orange")
> plot(dnorm, min(tsData[, 1]), max(tsData[, 1]), xlab = "Returns",
     yla = "Density", main = "Density")
> grid()
> plot(tsData, xlab = "Index", ylab = "Returns", main = "Return Series")
> abline(h = 0)
```



Function:	
hist	creates a histogram plot
truehist	plots a true histogram with density of total area one
density	computes and displays kernel density estimates
boxplot	produces a box-and-whisker plot
qqnorm	creates a normal quantile-quantile plot
qqline	adds a line through the first and third quartiles
qqplot	produces a QQ plot of two datasets

The foreign exchange rates can be downloaded from the FRED2 database of the US Federal Reserve Bank in St. Louis

```
> RATE <- "DEXUSEU"
> URL <- paste("http://research.stlouisfed.org/fred2/series/",
    RATE, "/", "downloaddata/", RATE, ".csv", sep = "")
> URL
[1] "http://research.stlouisfed.org/fred2/series/DEXUSEU/downloaddata/DEXUSEU.csv"
> USDEUR <- read.csv(URL)
> head(USDEUR)
        DATE  VALUE
1 1999-01-04 1.1812
2 1999-01-05 1.1760
3 1999-01-06 1.1636
4 1999-01-07 1.1672
5 1999-01-08 1.1554
6 1999-01-11 1.1534
```

The daily log-return vector is computed from the differences of the logarithms of the rates

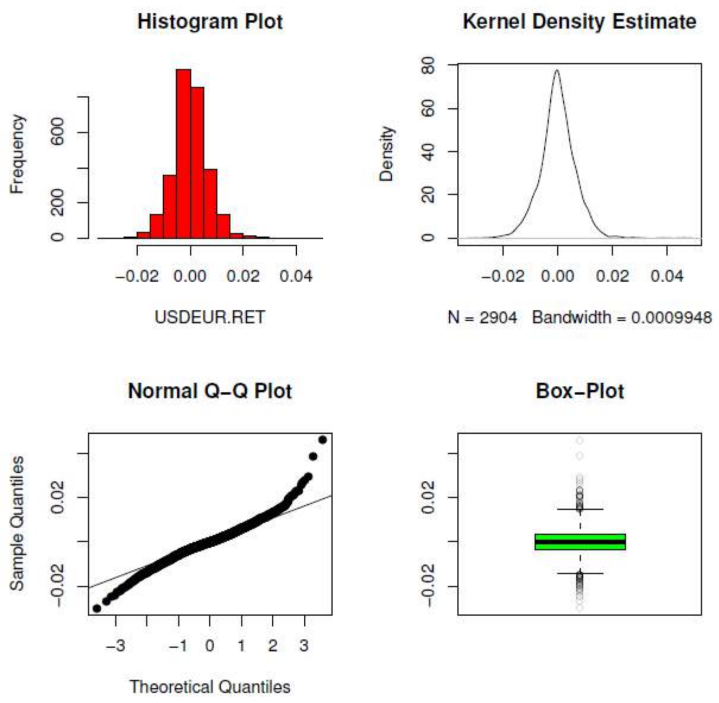
```
> USDEUR.RET = diff(log(USDEUR[, 2]))
```

```
> hist(USDEUR.RET, col = 2, main = "Histogram Plot")
```

```
> density = density(USDEUR.RET)
> plot(density, main = "Kernel Density Estimate")
```

```
> qqnorm(USDEUR.RET, pch = 19)
> qqline(USDEUR.RET)
```

```
> boxplot(USDEUR.RET, col = "green", main = "Box-Plot")
```



The functions pie() and barplot() can be used to draw pie and bar plots.

```
  > portfolioWeights = c(SwissBonds = 35, SwissEquities = 20, ForeignBonds = 25,
        ForeignEquities = 10, Commodities = 5, PrivateEquities = 5)
```

The following plot command creates a vertical view on the weights bars.

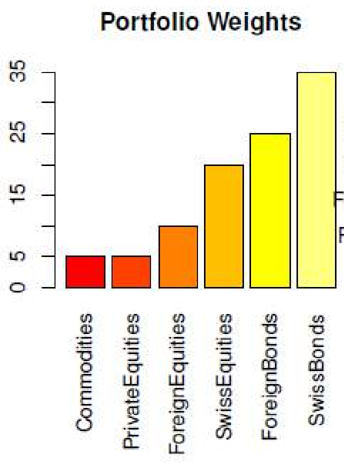
```
> barplot(sort(portfolioWeights), las = 3, col = heat.colors(6),
      offset = 0)
> title(main = "Portfolio Weights")
```

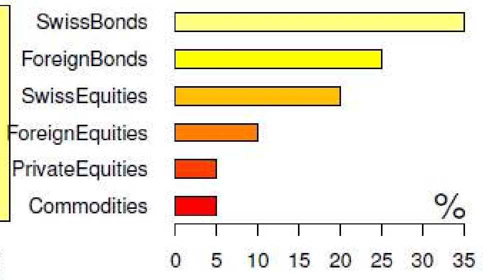
A horizontal view can be created by the following R command.

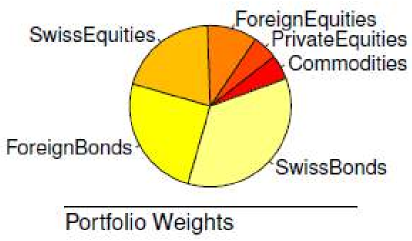
```
> barplot(sort(portfolioWeights), horiz = TRUE, las = 1, col = heat.colors(6),
      space = 1)
> text(33.3, 1.5, "%", cex = 1.8)
```

And the last example for the weights plot show how to create a pie plot.

```
> pie(sort(portfolioWeights), init.angle = 20, col = heat.colors(6))
> abline(h = -1)
> mtext(side = 1, line = -0.2, "Portfolio Weights", cex = 0.9,
      adj = 0)
```







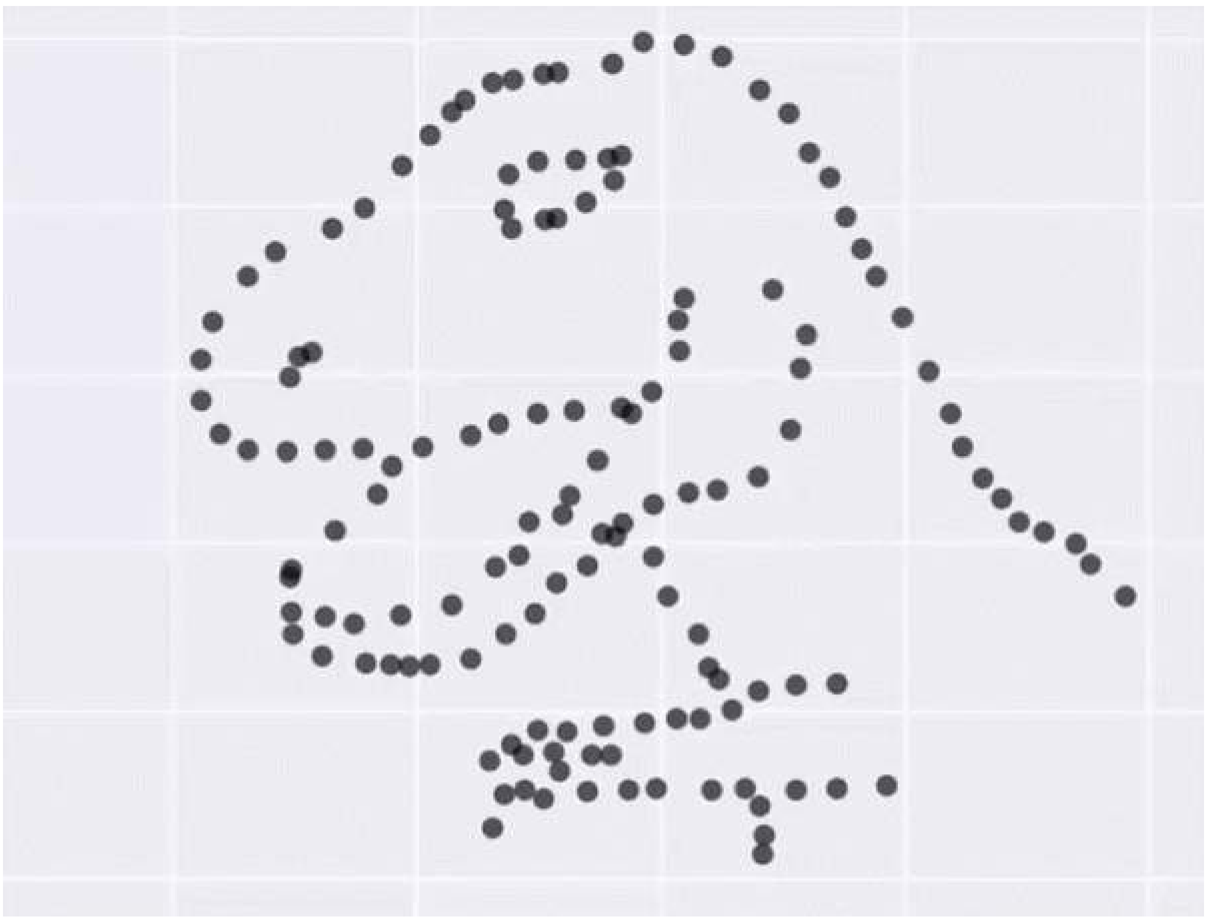
Introduction

Visualizing multi-dimensional data is an art as well as a science. Due to the limitations of our two-dimensional (2-D) rendering devices, building effective visualizations on more than two data dimensions (attributes or features) becomes challenging as the number of dimensions start increasing. We have extensively covered some strategies for effective multi-dimensional data visualization in one of my previous articles, *"The Art of Effective Visualization of Multi-dimensional Data"* with hands-on examples. In this article, we will cover the layered framework which we leveraged to build these visualizations, called *'The Grammar of Graphics'*. We will also explore essential concepts behind the layered Grammar of Graphics framework and discuss how we can use each specific layered component to build effective visualization on multi-dimensional data. Examples will be shown in Python, however, if you are interested you can replicate the same in R with ease.

Motivation

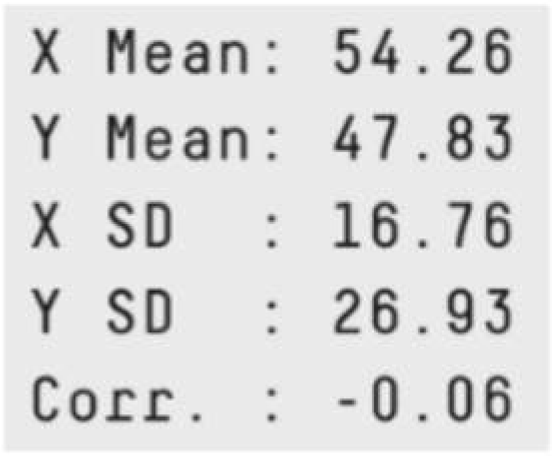
Data visualization and storytelling has always been one of the most important phases of any data science pipeline involving extracting meaningful insights from data, regardless of the complexity of the data or the project. Take a simple example of *'The Datasaurus Dozen'* — twelve different datasets depicted in the following figure.

Can you guess what is common among these very different looking sets of data?



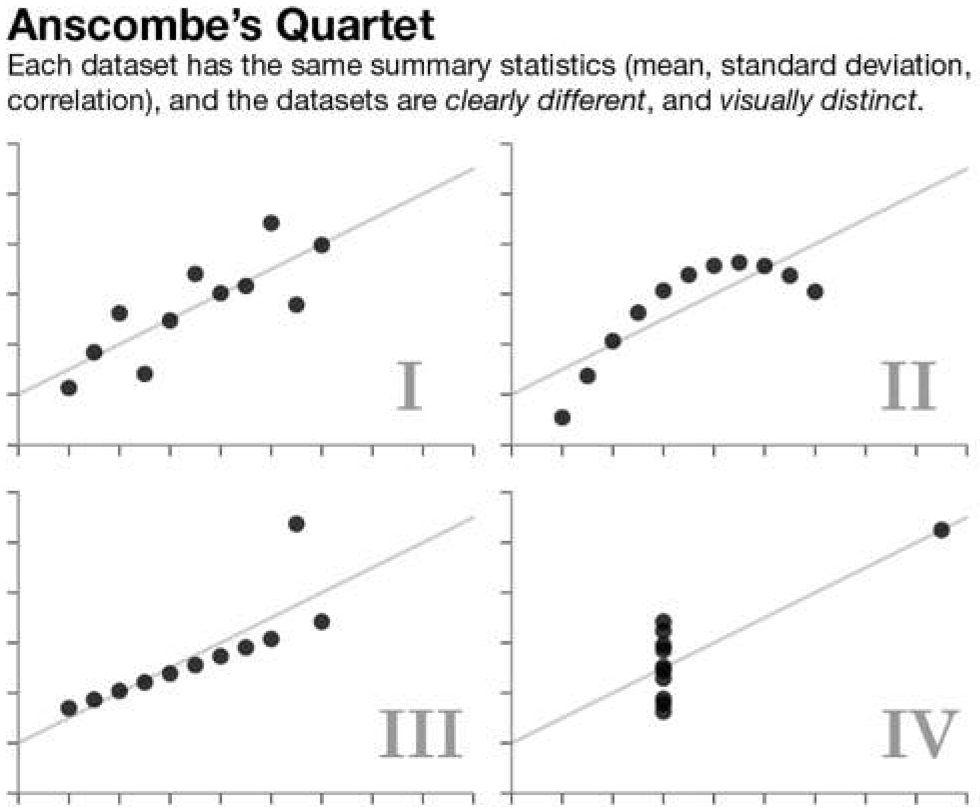
The Datasaurus Dozen — What is common among these diverse datasets?

Answer: Summary statistics for all the datasets are exactly the same!



```
X Mean: 54.26
Y Mean: 47.83
X SD  : 16.76
Y SD  : 26.93
Corr. : -0.06
```

This is a fun variant of the well known Anscombe's quartet, which many of you might be very familiar with, as depicted in the following figure.



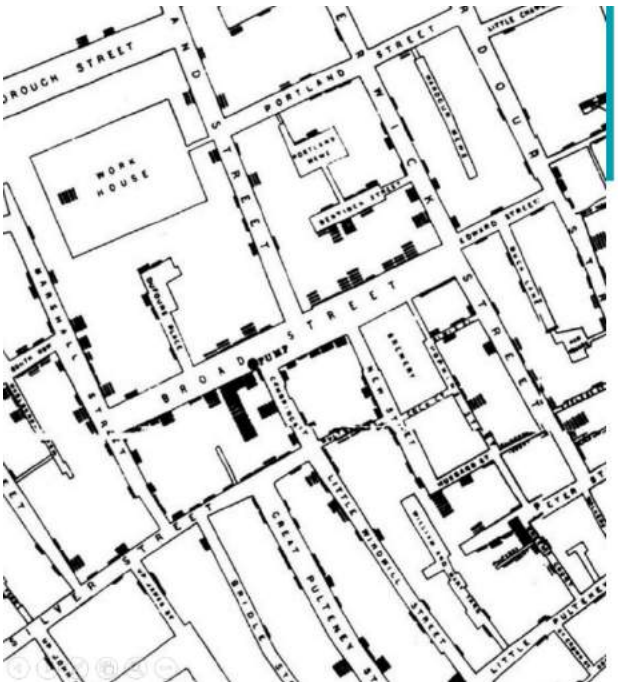
Anscombe's Quartet

Each dataset has the same summary statistics (mean, standard deviation, correlation), and the datasets are *clearly different*, and *visually distinct*.

Anscombe's quartet — Different datasets with same summary statistics

The key takeaway from these demonstrations would be, *“Do not trust your data blindly, and start modeling on your data”*. Summary statistics can always be deceptive. Always visualize and understand your data attributes before moving on to feature engineering and building statistical, machine learning and deep learning models.

Another very important source of motivation, particularly for effective data visualization, can be derived from some excellent case-studies dating several centuries back when we didn’t even have computers let alone Python or R! The first one is John Snow’s famous visualization depicting the Broad Street Cholera Outbreak in London, England in 1854!

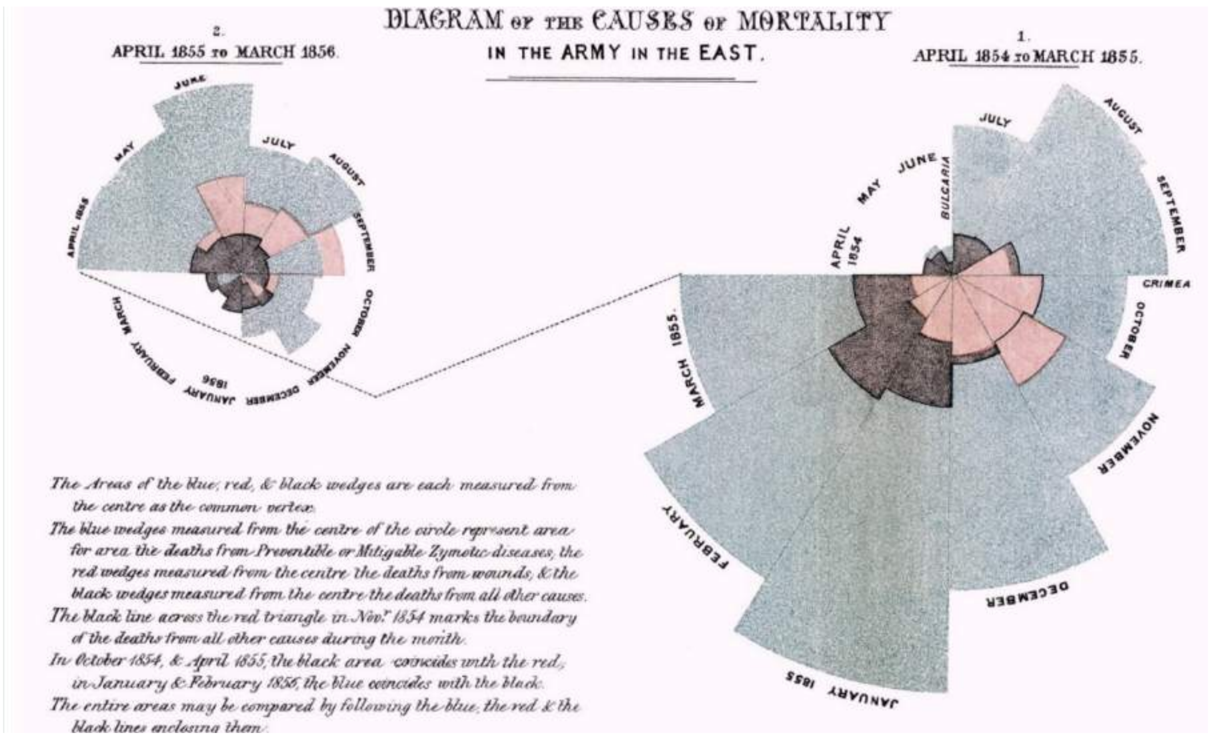


Visualizing the Broad Street Cholera Outbreak

- A severe outbreak of cholera occurred in 1854 near Broad Street in the City of Westminster, London, England, unknowing to people causing over 600 deaths
- Physician Jon Snow identified the source of the outbreak as the public water pump on Broad Street
- Snow used a dot map to illustrate the cluster of cholera cases around the pump
- He also used statistics to illustrate the connection between the quality of the water source and cholera cases.

Visualizing the Broad Street Cholera outbreak which helped find the root cause of the disease outbreak!

You can see how a simple hand-drawn visualization helped find the root cause of the cholera outbreak in Broad Street way back in the 1850s. Another interesting visualization was built by Florence Nightingale, the mother of modern nursing practice, who had a deep-seated interest in nursing and statistics.



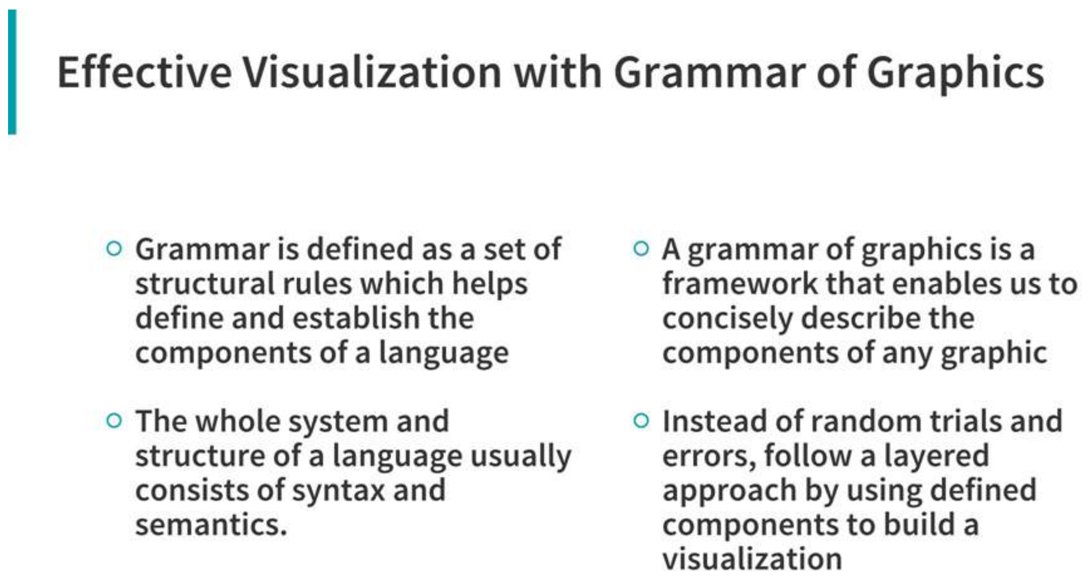
The Areas of the blue, red, & black wedges are each measured from the centre as the common vertex.
The blue wedges measured from the centre of the circle represent area for area the deaths from Preventible or Mitigable Zymotic diseases, the red wedges measured from the centre the deaths from wounds, & the black wedges measured from the centre the deaths from all other causes.
The black line across the red triangle in Novr. 1854 marks the boundary of the deaths from all other causes during the month.
In October 1854, & April 1855, the black area coincides with the red, in January & February 1856, the blue coincides with the black.
The entire areas may be compared by following the blue, the red & the black lines enclosing them.

Causes of Mortality in the Army of the East — Florence Nightingale

The above figure depicts a polar area diagram depicting causes of mortality (death) in the army in the 1850s. We can see the visualization is definitely not simplistic, yet it conveys the right insights — clearly showing the proportion of soldiers who died due to diseases which were preventable, based on wounds or other causes. This should serve as enough motivation for effective data visualization!

Understanding the Grammar of Graphics

To understand the Grammar of Graphics, we would need to understand what do we mean by Grammar. The following figure summarizes both these aspects briefly.



Effective Visualization with Grammar of Graphics

- Grammar is defined as a set of structural rules which helps define and establish the components of a language
- The whole system and structure of a language usually consists of syntax and semantics.
- A grammar of graphics is a framework that enables us to concisely describe the components of any graphic
- Instead of random trials and errors, follow a layered approach by using defined components to build a visualization

Basically, a grammar of graphics is a framework which follows a layered approach to describe and construct visualizations or graphics in a structured manner. A visualization involving multi-dimensional data often has multiple components or aspects, and leveraging this layered grammar of graphics helps us describe and understand each component involved in visualization — in terms of data, aesthetics, scale, objects and so on.

The original grammar of graphics framework was proposed by Leland Wilkinson, which covers all major aspects pertaining to effective data visualization in detail. I would definitely recommend interested readers to check out the book on it, whenever they get a chance!

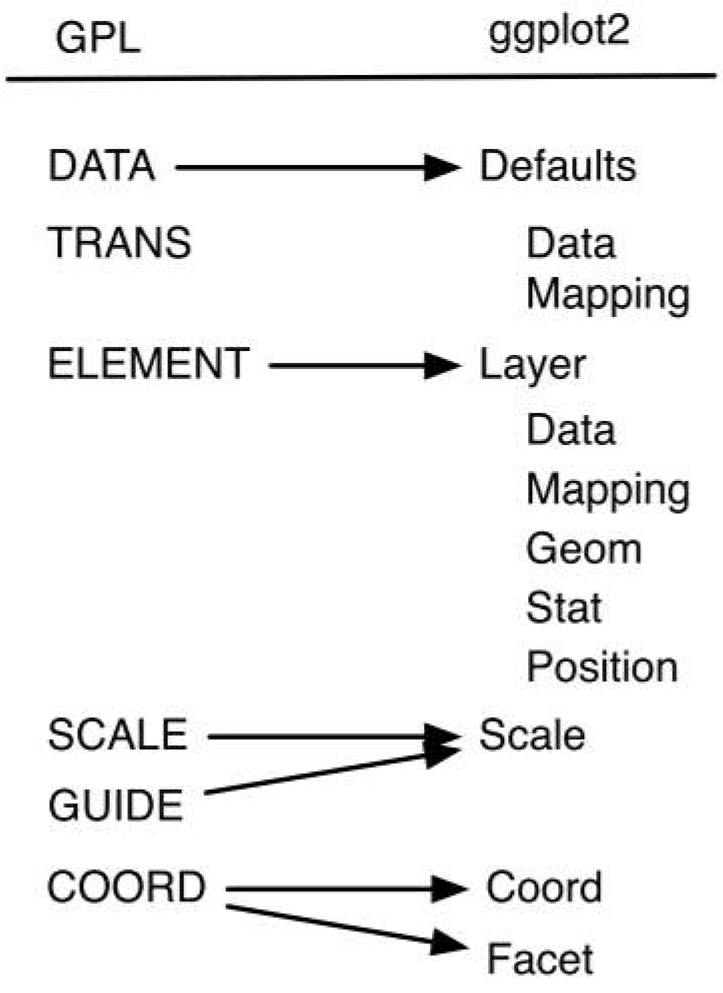
The Grammar of Graphics | Leland Wilkinson | Springer
Preface to First Edition Before writing the graphics for SYSTAT in the 1980's, I began by teaching a seminar in…
www.springer.com

We will, however, be using a variant of this — known as the layered grammar of graphics framework, which was proposed by Hadley Wickham, reputed Data Scientist and the creator of the famous R visualization package **ggplot2**. Readers should check out his paper titled, *'A layered grammar of graphics'* which covers his proposed layered grammar of graphics in detail and also talks about his open-source implementation framework **ggplot2** which was built for the R programming language

A layered grammar of graphics
A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar…

vita.had.co.nz

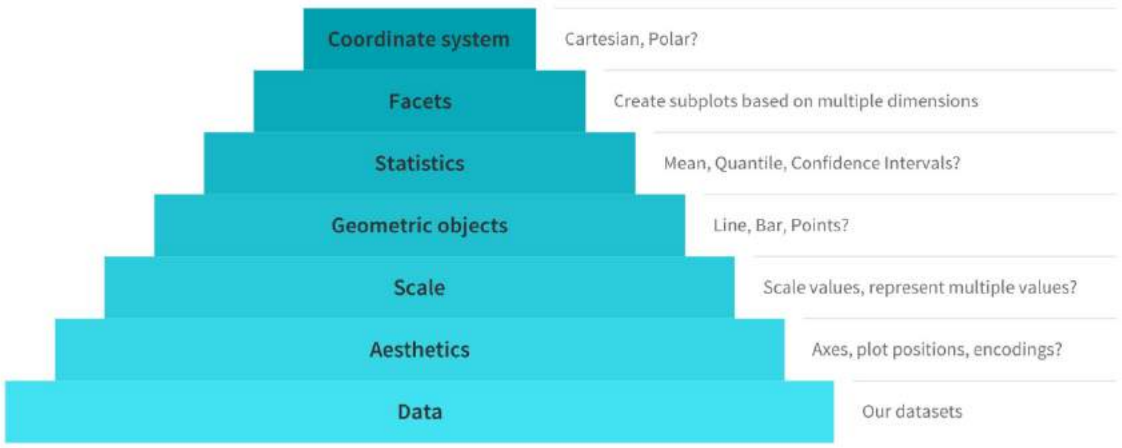
Hadley's layered grammar of graphics uses several layered components to describe any graphic or visualization. Most notably, it has some variations from the original grammar of graphics proposed by Wilkinson as depicted in the following figure.



Mapping between components of Wilkinson's grammar (left) and the layered grammar (right)

You can see there are various components in the grammar which can be used to build and describe a visualization. I have identified *seven* such major components which usually help me build effective visualizations on multi-dimensional data. The following figure illustrates it with some details about each specific component in the grammar.

Major Components of the Grammar of Graphics


Coordinate system
Cartesian, Polar?
Facets
Create subplots based on multiple dimensions
Statistics
Mean, Quantile, Confidence Intervals?
Geometric objects
Line, Bar, Points?
Scale
Scale values, represent multiple values?
Aesthetics
Axes, plot positions, encodings?
Data
Our datasets

Major components of the Grammar of Graphics

We illustrate the same using a pyramid architecture to show an inherent layered hierarchy of components. Typically, to build or describe any visualization with one or more dimensions, we can use the components as follows.

1. **Data**: Always start with the data, identify the dimensions you want to visualize.

2. **Aesthetics**: Confirm the axes based on the data dimensions, positions of various data points in the plot. Also check if any form of encoding is needed including size, shape, color and so on which are useful for plotting multiple data dimensions.

3. **Scale:** Do we need to scale the potential values, use a specific scale to represent multiple values or a range?

4. **Geometric objects:** These are popularly known as 'geoms'. This would cover the way we would depict the data points on the visualization. Should it be points, bars, lines and so on?

5. **Statistics:** Do we need to show some statistical measures in the visualization like measures of central tendency, spread, confidence intervals?

6. **Facets:** Do we need to create subplots based on specific data dimensions?

7. **Coordinate system:** What kind of a coordinate system should the visualization be based on — should it be cartesian or polar?

We will now be looking at how to leverage this layered framework to build effective data visualizations for multi-dimensional data with some hands-on examples.

Grammar of Graphics in Action

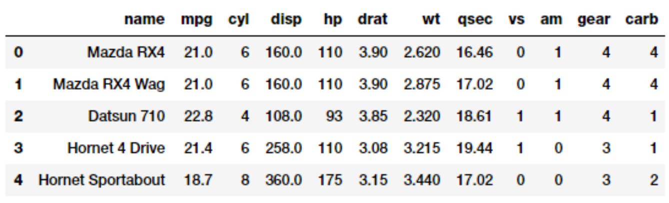
We will now apply the concept we learnt in the previous section on some actual data. We will be building all our visualizations in Python, but I would also recommend people to check out the **ggplot2** package in R, which is one of the most inspiring frameworks till now for building nice and clean publication quality visualizations. We do have a **ggplot** framework in Python which has been built by Yhat, Inc. However, if you check the repository, no commits or updates have been pushed for the last two years, and unfortunately, this causes some errors especially due to backward incompatibility issues with newer versions of **pandas**. Hence, to emulate the true layered grammar of graphics syntax in Python, we will use another interesting framework called **plotnine**.

A Grammar of Graphics for Python - plotnine 0.4.0 documentation
plotnine is an implementation of a grammar of graphics in Python, it is based on ggplot2. The grammar allows users to…
plotnine.readthedocs.io

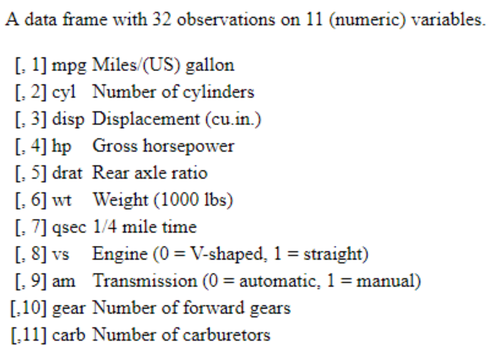
Plotnine is an open-source Python implementation for a layered grammar of graphics framework which is based on **ggplot2**. Thus, using the previously specified components in this layered grammar, we can build effective visualizations.

Start with the data

We always start by loading up and looking at the dataset we want to analyze and visualize. We will use the famous **mtcars** dataset available as one of the pre-loaded datasets in **plotnine**.



	name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2



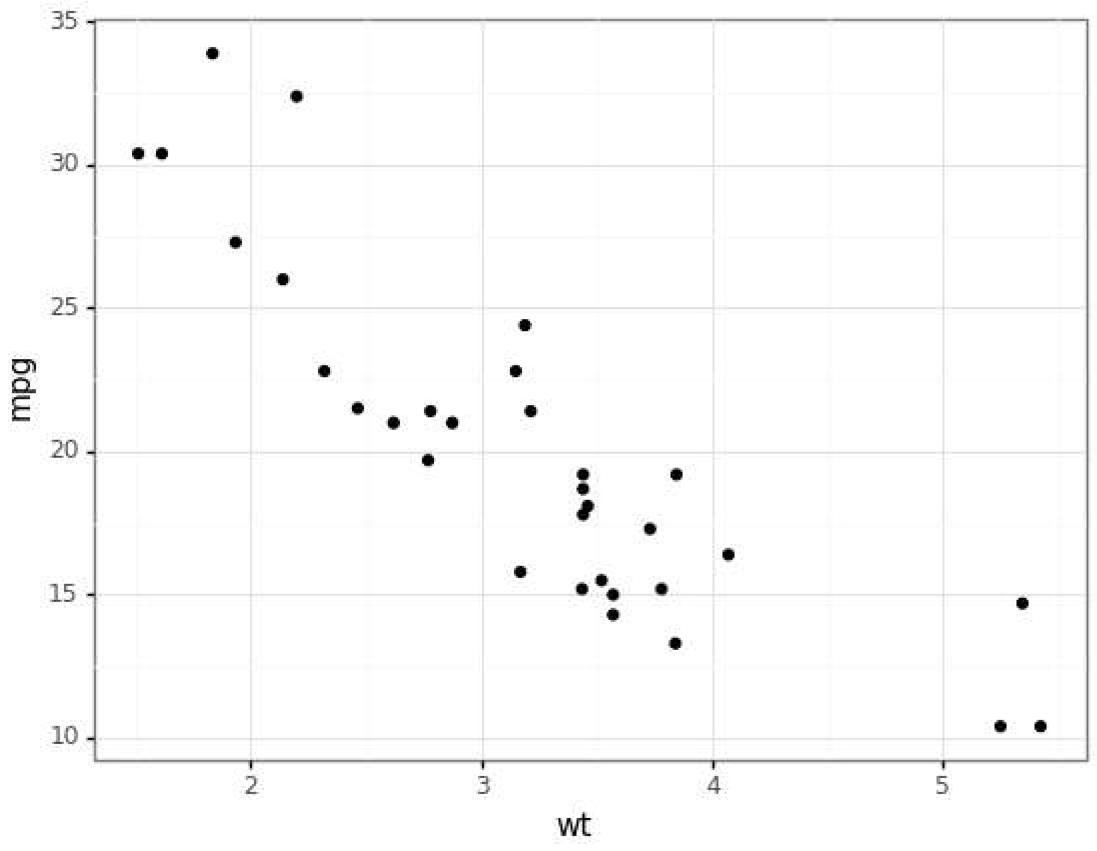
A data frame with 32 observations on 11 (numeric) variables.

[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (1000 lbs)
[, 7]	qsec	1/4 mile time
[, 8]	vs	Engine (0 = V-shaped, 1 = straight)
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	carb	Number of carburetors

The mtcars dataset consists of data that was extracted from the 1974 *Motor Trend* US magazine, and depicts fuel consumption and 10 other attributes of automobile design and performance for 32 automobiles (1973–74 models). The details of each attribute are depicted in the figure above. Let's build some visualizations now.

Visualize two-dimensions (2-D)

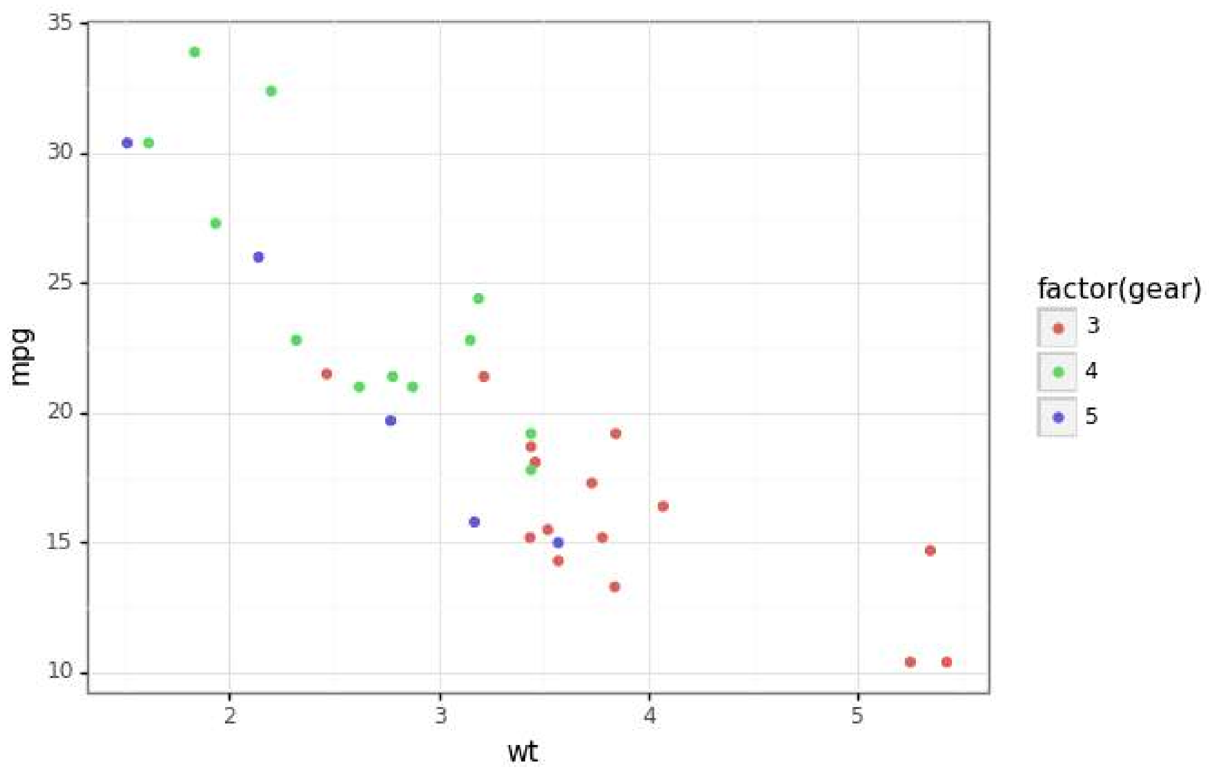
We can now visualize data up to two dimensions using some of the components from our layered grammar of graphics framework including data, scale, aesthetics and geoms. We choose a dot or scatter plot in this case for our geometric object to represent each data point.



We can clearly see from the above visualization that **mpg** has a negative correlation with the cat **wt**.

Visualize three-dimensions (3-D)

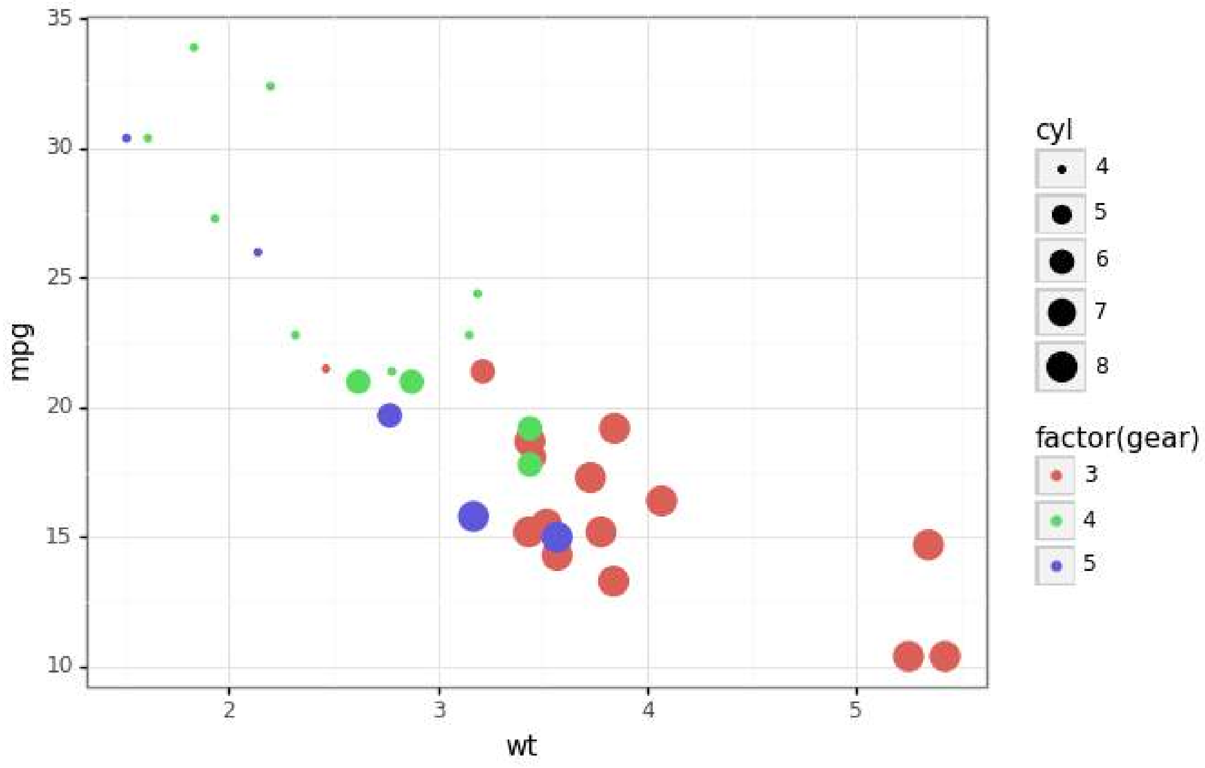
To visualize three dimensions from our dataset, we can leverage color as one of our aesthetic components to visualize one additional dimension besides our other two dimensions as depicted in the following example.



In the above visualization, we depict the cars with different number of gears as separate categories using the color aesthetic along with our other two data dimensions (variables). It is quite clear that cars with a smaller number of **gears** on average tend to have higher **wt** and lower **mpg**.

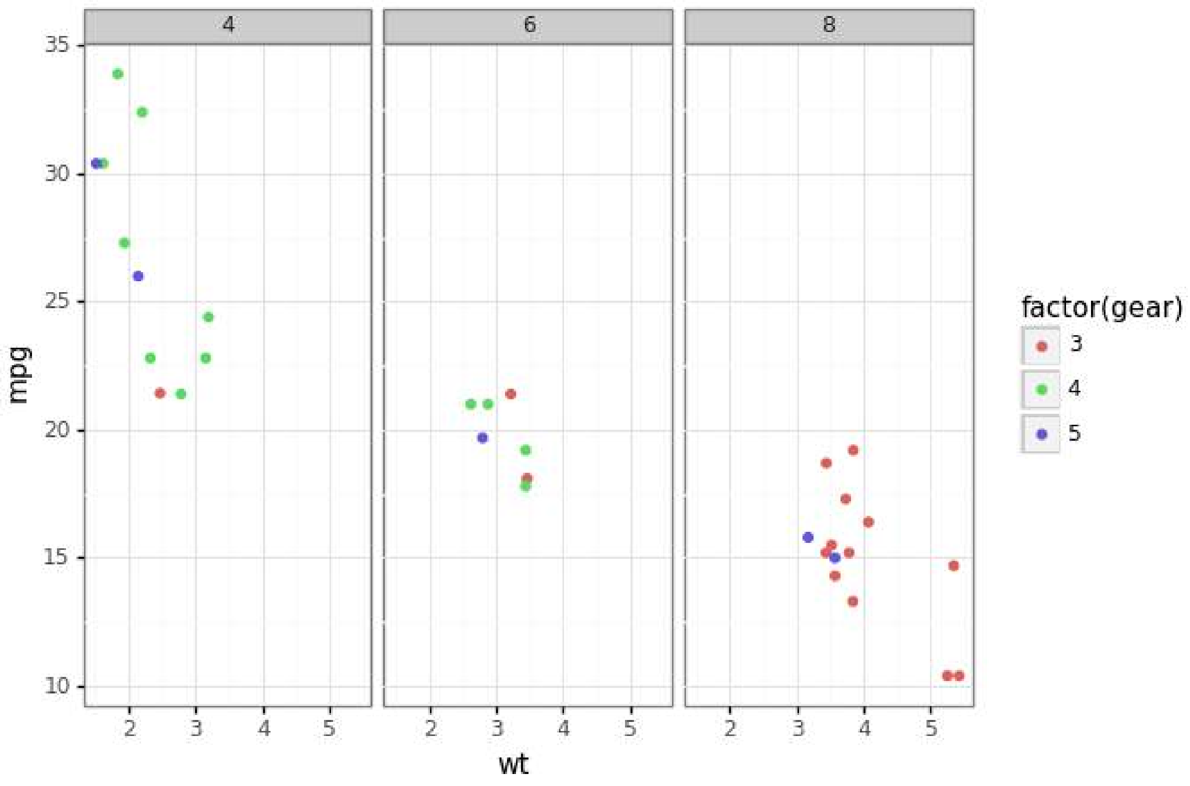
Visualize four-dimensions (4-D)

To visualize four dimensions from our dataset, we can leverage color as well as size as two of our aesthetics besides other regular components including geoms, data and scale.



The visualization shows us how powerful aesthetics can be in helping us visualize multiple data dimensions in a single plot. It is quite clear that cars with higher number of **cyl** (cylinders) have lower number of **gears** and in turn, their **wt** is higher and **mpg** lower.

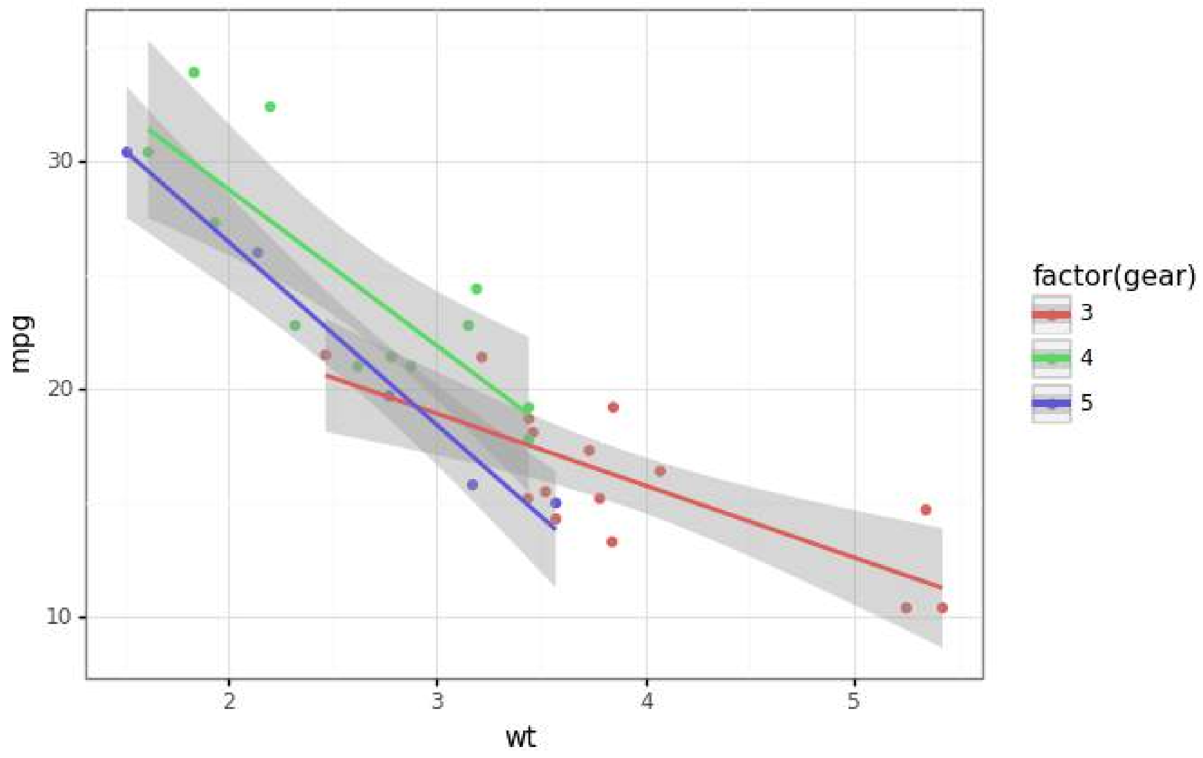
Alternatively, we can also use color and facets to depict data in four dimensions instead of size as depicted in the following example.



Facets are definitely one of the most powerful components towards building an effective data visualization as shown in the visualization above, where we can clearly see cars with higher **cyl** count have lower **gear** count and similar trends as the previous visualization with color and size.

Visualize data dimensions and statistics

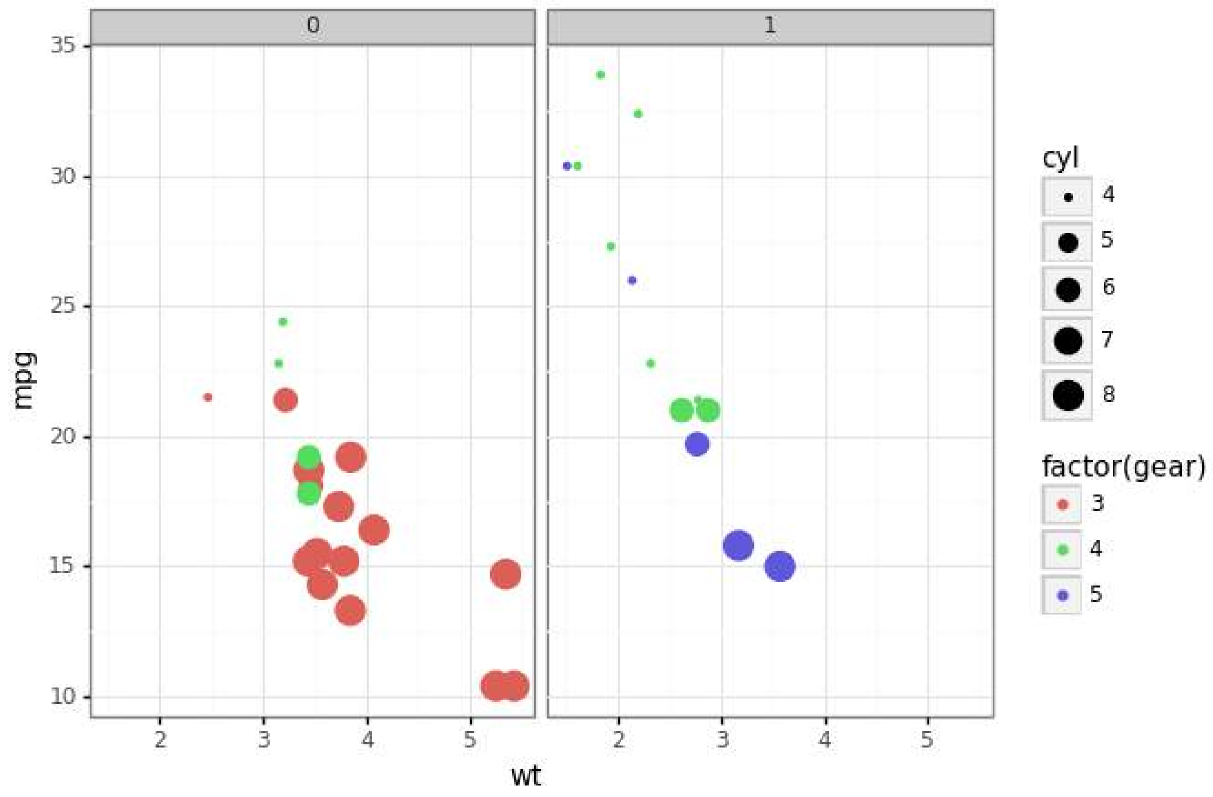
To visualize data dimensions and some relevant statistics (like fitting a linear model), we can leverage statistics along with the other components in our layered grammar.



This enables us to see linear model trends for **mpg** based on **wt** due to the statistics component.

Visualize five-dimensions (5-D)

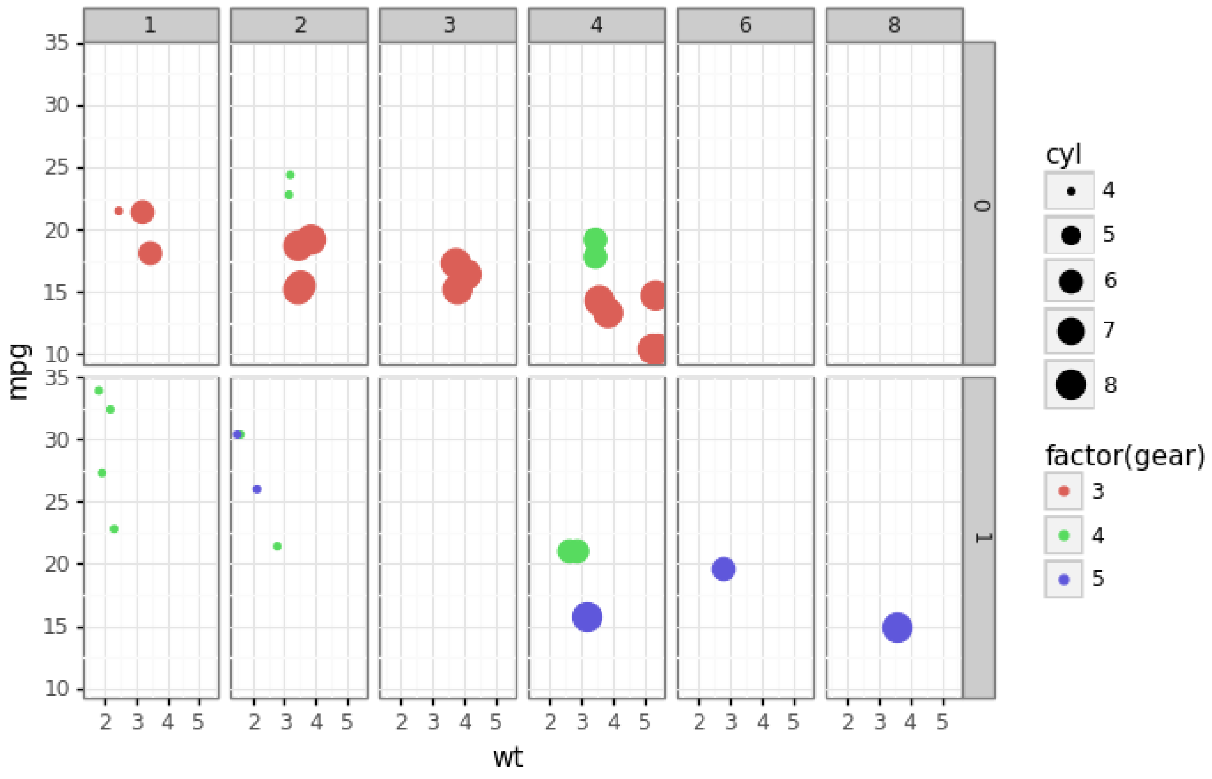
To visualize data in five dimensions, you know the drill by now! We will leverage the power of aesthetics including color, size and facets.



Here we use **am** as the facet where 0 indicates cars with automatic transmission and 1 indicates cars with manual transmission. The plot shows that cars with manual transmission have higher number of gears as compared to cars with automatic transmission. Also, majority of cars with a higher number of cylinders (**cyl**) have automatic transmission. Other insights are similar to what we have observed in the previous plots.

Visualize six-dimensions (6-D)

To visualize data in six dimensions, we can add in an additional facet on the *y-axis* along with a facet on the *x-axis*, and color and size as aesthetics.

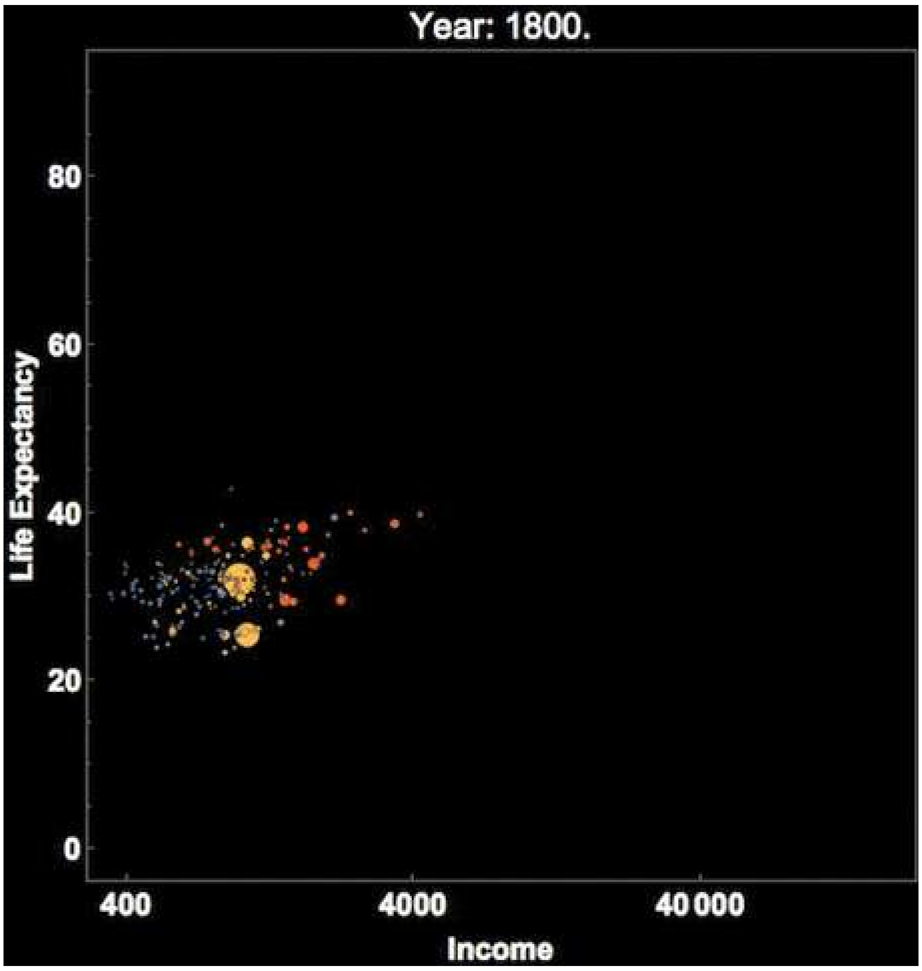


We represent transmission (**am**) as 0 (automatic) and 1 (manual) as a facet on the *y-axis* and number of carburetors (**carb**) as a facet on the *x-axis* besides the other dimensions represented using other aesthetics similar to our previous plots. An interesting insight from the above visualization is that cars with higher number of gears have manual transmission (**am**) and higher number of carburetors (**carb**). Do you notice any other interesting insights?

Can we go higher?

The pressing question is, can we go higher than six dimensions? Well, it definitely becomes more and more difficult to hack our way around the limitations of a two-dimensional rendering device to visualize more data dimensions.

One method is to use more facets and subplots. Besides this, you can also use the notion of time if your dataset has a temporal aspect as depicted in the following example.



Hans Rosling’s famous visualization of global population, health and economic indicators

This depicts *Hans Rosling’s* famous visualization of depicting global population, health and various economic indicators across all countries. This was also presented in an **official TED conference** which I would recommend everyone to check out if they haven’t done it already!

Laboratory Procotol Developer and Supervisor(s) Information
Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   oo. If you have any troubleshoot please contact your supervisor or Building Management
   pp. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   qq. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   rr. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 11

Date : November 12th, 2019

Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 9, students are expected to understand how to run an Application of data manipulation as it is about fitting models to data.

Procedure

Regression Analysis

Code for doing Regression Analysis

```
rm(list=ls())           # Clear R's memory

data(cars)              #call the data
head(cars)  # display the first 6 observations
plot(cars)
scatter.smooth(x=cars$speed, y=cars$dist, main="Dist ~ Speed")  #
scatterplot

par(mfrow=c(1, 2))  # divide graph area in 2 columns

boxplot(cars$speed, main="Speed", sub=paste("Outlier rows: ",
boxplot.stats(cars$speed)$out))  # box plot for 'speed'

boxplot(cars$dist, main="Distance", sub=paste("Outlier rows: ",
boxplot.stats(cars$dist)$out))  # box plot for 'distance'
```

```
#library(e1071)
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(cars$speed), main="Density Plot: Speed",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(cars$speed), 2)))  # density plot for 'speed'
polygon(density(cars$speed), col="red")
plot(density(cars$dist), main="Density Plot: Distance",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(cars$dist), 2)))  # density plot for 'dist'
polygon(density(cars$dist), col="red")

cor(cars$speed, cars$dist)  # calculate correlation between speed
and distance
linearMod <- lm(dist ~ speed, data=cars)  # build linear regression
model on full data
print(linearMod)
summary(linearMod)
anova(linearMod)

plot(linearMod, which=1)
plot(linearMod, which=2)
plot(linearMod, which=3)
plot(linearMod, which=4)
```

`rm(list=ls())`	: is a code to remove all data that is called or inputed in the Rstudio
`data()`	: is to call data provided in R
`plot()`	: is to create scatter plot of the data

`scatter.smooth(x, y, main="")`: is to create a smoothing line for data plotted
- x : filled by variable put in x axis
- y : filled by variable put in y axis
- main : is a code to give a title for the plot

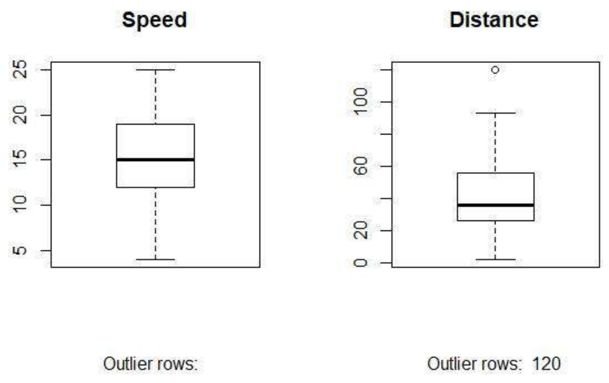
`boxplot()`	: to create a boxplot
`cor(a,b)`	: to calculate correlation between two variables a and b
`lm(y ~ x, data)`	: a function to create a regression model for variable independent (x) and variable dependent (y)
`summary(model)`	: is a code to write the summary and the test of regression model
`anova(linearMod)`	: is a code to do F test anaylsis
`print(linearMod`	: is a code to print the coefficient of the regression

`plot(linearMod, which=1)` : a code to create a error scatterplot for checking the regression form

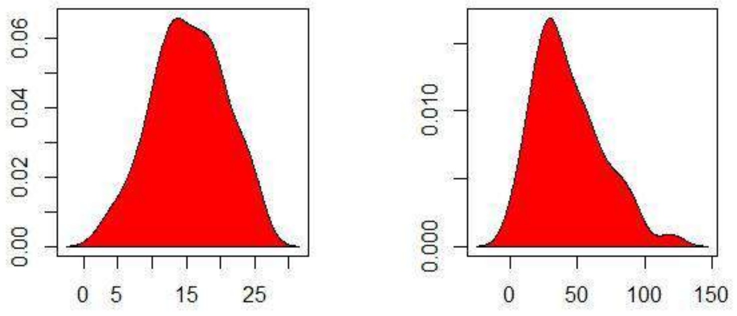
`par(mfrow=c(`*`mrows`*`,`*`ncols`*`))`: a code to create a matrix of mrows and ncols plots that are filled in by row.

Output

```
> plot(cars)
> scatter.smooth(x=cars$speed, y=cars$dist, main="Dist ~ Speed")  # 
scatterplot
> par(mfrow=c(1, 2))  # divide graph area in 2 columns
> boxplot(cars$speed, main="Speed", sub=paste("Outlier rows: ", 
boxplot.stats(cars$speed)$out))  # box plot for 'speed'
> boxplot(cars$dist, main="Distance", sub=paste("Outlier rows: ", 
boxplot.stats(cars$dist)$out))  # box plot for 'distance'
```



```
> plot(density(cars$speed), main="Density Plot: Speed", ylab="Frequency", 
sub=paste("Skewness:", round(e1071::skewness(cars$speed), 2)))  # density 
plot for 'speed'
> polygon(density(cars$speed), col="red")
> plot(density(cars$dist), main="Density Plot: Distance", ylab="Frequency", 
sub=paste("Skewness:", round(e1071::skewness(cars$dist), 2)))  # density 
plot for 'dist'
> polygon(density(cars$dist), col="red")
```



```
> cor(cars$speed, cars$dist)  # calculate correlation between speed and 
distance
[1] 0.8068949
> linearMod <- lm(dist ~ speed, data=cars)  # build linear regression model 
on full data
> linearMod
```

```
Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)        speed
    -17.579        3.932


> summary(linearMod)

Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,   Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12

> anova(linearMod)
Analysis of Variance Table

Response: dist
          Df Sum Sq Mean Sq F value   Pr(>F)
speed      1  21186 21185.5  89.567 1.49e-12 ***
Residuals 48  11354   236.5
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   ss. If you have any troubleshoot please contact your supervisor or Building Management
   tt. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   uu. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   vv. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 12

Date : December 3rd, 2019
Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 9, students are expected to understand how to run an Application of data manipulation as it is about fitting models to data.

Procedure

R has five basic or "atomic" classes of objects:

- character
- numeric (real numbers)
- integer
- complex
- logical (True/False)

Creating Vectors

The `c()` function can be used to create vectors of objects by concatenating things together.

```
x <- c(0.5, 0.6)       ## numeric
x <- c(TRUE, FALSE)    ## logical
x <- c(T, F)           ## logical
x <- c("a", "b", "c")  ## character
x <- 9:29              ## integer
x <- c(1+0i, 2+4i)     ## complex
```

You can also use the `vector()` function to initialize vectors.

```
x <- vector("numeric", length = 10)
```

Mixing Objects

There are occasions when different classes of R objects get mixed together. Sometimes this happens by accident but it can also happen on purpose. So what happens with the following code?

```
y <- c(1.7, "a")   ## character
y <- c(TRUE, 2)    ## numeric
y <- c("a", TRUE)  ## character
```

Explicit Coercion

Objects can be explicitly coerced from one class to another using the `as.*` functions, if available.

```
x <- 0:6
class(x)
[1] "integer"
as.numeric(x)
[1] 0 1 2 3 4 5 6
as.logical(x)
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
as.character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

Sometimes, R can’t figure out how to coerce an object and this can result in `NA`s being produced.

```
x <- c("a", "b", "c")
as.numeric(x)
Warning: NAs introduced by coercion
[1] NA NA NA
as.logical(x)
[1] NA NA NA
as.complex(x)
Warning: NAs introduced by coercion
```

Matrices

Matrices are vectors with a *dimension* attribute. The dimension attribute is itself an integer vector of length 2 (number of rows, number of columns)

```
m <- matrix(nrow = 2, ncol = 3)
m
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
dim(m)
[1] 2 3
attributes(m)
$dim
[1] 2 3
```

Matrices are constructed *column-wise*, so entries can be thought of starting in the “upper left” corner and running down the columns.

```
m <- matrix(1:6, nrow = 2, ncol = 3)
m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Matrices can also be created directly from vectors by adding a dimension attribute.

```
m <- 1:10
m
 [1]  1  2  3  4  5  6  7  8  9 10
dim(m) <- c(2, 5)
m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

Matrices can be created by *column-binding* or *row-binding* with the `cbind()` and `rbind()` functions.

```
x <- 1:3
y <- 10:12
cbind(x, y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

Lists

Lists are a special type of vector that can contain elements of different classes. Lists are a very important data type in R and you should get to know them well. Lists, in combination with the various "apply" functions discussed later, make for a powerful combination.

Lists can be explicitly created using the `list()` function, which takes an arbitrary number of arguments.

```
x <- list(1, "a", TRUE, 1 + 4i)
x
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```

We can also create an empty list of a prespecified length with the `vector()` function

```
x <- vector("list", length = 5)
x
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL
```

Factors

Factors are used to represent categorical data and can be unordered or ordered. One can think of a factor as an integer vector where each integer has a *label*. Factors are important in statistical modeling and are treated specially by modelling functions like `lm()` and `glm()`.
Using factors with labels is *better* than using integers because factors are self-describing. Having a variable that has values "Male" and "Female" is better than a variable that has values 1 and 2.

Factor objects can be created with the `factor()` function.

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
[1] yes yes no  yes no
Levels: no yes
table(x)
x
 no yes
  2   3
## See the underlying representation of factor
unclass(x)
[1] 2 2 1 2 1
attr(,"levels")
[1] "no"  "yes"
```

Often factors will be automatically created for you when you read a dataset in using a function like `read.table()`. Those functions often, as a default, create factors when they encounter data that look like characters or strings.
The order of the levels of a factor can be set using the `levels` argument to `factor()`. This can be important in linear modelling because the first level is used as the baseline level. This feature can also be used to customize order in plots that include factors, since by default factors are plotted in the order of their levels.

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x  ## Levels are put in alphabetical order
[1] yes yes no  yes no
Levels: no yes
x <- factor(c("yes", "yes", "no", "yes", "no"),
            levels = c("yes", "no"))
x
[1] yes yes no  yes no
Levels: yes no
```

Missing Values

Missing values are denoted by `NA` or `NaN` for undefined mathematical operations.

- `is.na()` is used to test objects if they are `NA`
- `is.nan()` is used to test for `NaN`
- `NA` values have a class also, so there are integer `NA`, character `NA`, etc.
- A `NaN` value is also `NA` but the converse is not true

```
## Create a vector with NAs in it
x <- c(1, 2, NA, 10, 3)
## Return a logical vector indicating which elements are NA
is.na(x)
[1] FALSE FALSE  TRUE FALSE FALSE
## Return a logical vector indicating which elements are NaN
is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE
## Now create a vector with both NA and NaN values
x <- c(1, 2, NaN, NA, 4)
```

```
is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE
is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE
```

Data Frames

Data frames are used to store tabular data in R. They are an important type of object in R and are used in a variety of statistical modeling applications. Hadley Wickham's package dplyr has an optimized set of functions designed to work efficiently with data frames, and `ggplot2` plotting functions work best with data stored in data frames.
Data frames are represented as a special type of list where every element of the list has to have the same length. Each element of the list can be thought of as a column and the length of each element of the list is the number of rows.

Unlike matrices, data frames can store different classes of objects in each column. Matrices must have every element be the same class (e.g. all integers or all numeric).

In addition to column names, indicating the names of the variables or predictors, data frames have a special attribute called `row.names` which indicate information about each row of the data frame.
Data frames are usually created by reading in a dataset using the `read.table()` or `read.csv()`. However, data frames can also be created explicitly with the `data.frame()` function or they can be coerced from other types of objects like lists.
Data frames can be converted to a matrix by calling `data.matrix()`. While it might seem that the `as.matrix()` function should be used to coerce a data frame to a matrix, almost always, what you want is the result of `data.matrix()`.

```
x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
x
  foo   bar
1   1  TRUE
2   2  TRUE
3   3 FALSE
4   4 FALSE
nrow(x)
[1] 4
ncol(x)
[1] 2
```

Names

R objects can have names, which is very useful for writing readable code and self-describing objects. Here is an example of assigning names to an integer vector.

```
x <- 1:3
names(x)
NULL
names(x) <- c("New York", "Seattle", "Los Angeles")
x
   New York     Seattle Los Angeles
          1           2           3
names(x)
[1] "New York"    "Seattle"     "Los Angeles"
```

Lists can also have names, which is often very useful.

```
x <- list("Los Angeles" = 1, Boston = 2, London = 3)
```

```
x
$`Los Angeles`
[1] 1

$Boston
[1] 2

$London
[1] 3
names(x)
[1] "Los Angeles" "Boston"      "London"
```

Matrices can have both column and row names.

```
m <- matrix(1:4, nrow = 2, ncol = 2)
dimnames(m) <- list(c("a", "b"), c("c", "d"))
m
  c d
a 1 3
b 2 4
```

Column names and row names can be set separately using the `colnames()` and `rownames()` functions.

```
colnames(m) <- c("h", "f")
rownames(m) <- c("x", "z")
m
  h f
x 1 3
z 2 4
```

Note that for data frames, there is a separate function for setting the row names, the `row.names()` function. Also, data frames do not have column names, they just have names (like lists). So to set the column names of a data frame just use the `names()` function. Yes, I know its confusing. Here's a quick summary:

Object	Set column names	Set row names
data frame	`names()`	`row.names()`
matrix	`colnames()`	`rownames()`

Reading Tabular Data with the `readr` Package
The learning objectives for this section are to:

- Read tabular data into R and read in web data via web scraping tools and APIs

The `readr` package is the primary means by which we will read tablular data, most notably, comma-separated-value (CSV) files. The `readr` package has a few functions in it for reading and writing tabular data—we will focus on the `read_csv` function. The `readr` package is available on CRAN and the code for the package is maintained on GitHub.
The importance of the `read_csv` function is perhaps better understood from an historical perspective. R's built in `read.csv` function similarly reads CSV files, but the `read_csv` function in `readr` builds on that by removing some of the quirks and "gotchas" of `read.csv` as well as dramatically optimizing the speed with which it can read data into R. The `read_csv` function also adds some nice user-oriented features like a progress meter and a compact method for specifying column types.

The only required argument to `read_csv` is a character string specifying the path to the file to read. A typical call to `read_csv` will look as follows.

```
library(readr)
teams <- read_csv("directory/file_name.csv")
```

The `readr` package includes a variety of functions in the `read_*` family that allow you to read in data from different formats of flat files. The following table gives a guide to several functions in the `read_*` family.

`readr` function	**Use**
`read_csv`	Reads comma-separated file
`read_csv2`	Reads semicolon-separated file
`read_tsv`	Reads tab-separated file
`read_delim`	General function for reading delimited files
`read_fwf`	Reads fixed width files
`read_log`	Reads log files

Control Structures

Note: Some of the material in this section is adapted from R Programming for Data Science.

The learning objectives of the section are:

- Describe the control flow of an R program

Control structures in R allow you to control the flow of execution of a series of R expressions. Basically, control structures allow you to put some "logic" into your R code, rather than just always executing the same R code every time. Control structures allow you to respond to inputs or to features of the data and execute different R expressions accordingly.

Commonly used control structures are

- `if` and `else`: testing a condition and acting on it
- `for`: execute a loop a fixed number of times
- `break`: break the execution of a loop
- `next`: skip an iteration of a loop

Most control structures are not used in interactive sessions, but rather when writing functions or longer expressions. However, these constructs do not have to be used in functions and it's a good idea to become familiar with them before we delve into functions.

`if-else`

The `if-else` combination is probably the most commonly used control structure in R (or perhaps any language). This structure allows you to test a condition and act on it depending on whether it's true or false.

For starters, you can just use the `if` statement.

```
if(<condition>) {
        ## do something
}
## Continue with rest of code
```

The above code does nothing if the condition is false. If you have an action you want to execute when the condition is false, then you need an `else` clause.

```
if(<condition>) {
        ## do something
} else {
        ## do something else
}
```

You can have a series of tests by following the initial `if` with any number of `else ifs`.

```
if(<condition1>) {
        ## do something
} else if(<condition2>)  {
        ## do something different
} else {
        ## do something different
}
```

Here is an example of a valid if/else structure.

```
## Generate a uniform random number
x <- runif(1, 0, 10)
if(x > 3) {
        y <- 10
} else {
        y <- 0
}
```

The value of `y` is set depending on whether `x > 3` or not.
Of course, the `else` clause is not necessary. You could have a series of if clauses that always get executed if their respective conditions are true.

```
if(<condition1>) {

}

if(<condition2>) {

}
```

`for` Loops
For loops are pretty much the only looping construct that you will need in R. While you may occasionally find a need for other types of loops, in most data analysis situations, there are very few cases where a for loop isn't sufficient.

In R, for loops take an iterator variable and assign it successive values from a sequence or vector. For loops are most commonly used for iterating over the elements of an object (list, vector, etc.)

```
numbers <- rnorm(10)
for(i in 1:10) {
        print(numbers[i])
}
[1] -0.9567815
[1] 1.347491
[1] -0.03158058
[1] 0.5960358
[1] 1.133312
[1] -0.7085361
[1] 1.525453
[1] 1.114152
[1] -0.1214943
[1] -0.2898258
```

This loop takes the `i` variable and in each iteration of the loop gives it values 1, 2, 3, …, 10, executes the code within the curly braces, and then the loop exits.
The following three loops all have the same behavior.

```
x <- c("a", "b", "c", "d")

for(i in 1:4) {
        ## Print out each element of 'x'
        print(x[i])
}
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

The `seq_along()` function is commonly used in conjunction with for loops in order to generate an integer sequence based on the length of an object (in this case, the object x).

```
## Generate a sequence based on length of 'x'
for(i in seq_along(x)) {
        print(x[i])
}
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

It is not necessary to use an index-type variable.

```
for(letter in x) {
        print(letter)
}
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

For one line loops, the curly braces are not strictly necessary.

```
for(i in 1:4) print(x[i])
[1] "a"
[1] "b"
[1] "c"
[1] "d"
```

However, curly braces are sometimes useful even for one-line loops, because that way if you decide to expand the loop to multiple lines, you won't be burned because you forgot to add curly braces (and you *will* be burned by this).

Nested `for` loops
`for` loops can be nested inside of each other.

```
x <- matrix(1:6, 2, 3)

for(i in seq_len(nrow(x))) {
        for(j in seq_len(ncol(x))) {
                print(x[i, j])
        }
}
```

Nested loops are commonly needed for multidimensional or hierarchical data structures (e.g. matrices, lists). Be careful with nesting though. Nesting beyond 2 to 3 levels often makes it difficult to read or understand the code. If you find yourself in need of a large number of nested loops, you may want to break up the loops by using functions (discussed later).

`next`, `break`
`next` is used to skip an iteration of a loop.
```
for(i in 1:100) {
        if(i <= 20) {
                ## Skip the first 20 iterations
                next
        }
        ## Do something here
}
```
`break` is used to exit a loop immediately, regardless of what iteration the loop may be on.
```
for(i in 1:100) {
      print(i)

      if(i > 20) {
              ## Stop loop after 20 iterations
              break
      }
}
```

Basic Plotting With ggplot2

The `ggplot2` package allows you to quickly plot attractive graphics and to visualize and explore data. Objects created with `ggplot2` can also be extensively customized with `ggplot2` functions (more on that in the next subsection), and because `ggplot2` is built using grid graphics, anything that cannot be customized using `ggplot2` functions can often be customized using grid graphics. While the structure of `ggplot2` code differs substantially from that of base R graphics, it offers a lot of power for the required effort. This first subsection focuses on **useful**, rather than **attractive** graphs, since this subsection focuses on exploring rather than presenting data. Later sections will give more information about making more attractive or customized plots, as you'd want to do for final reports, papers, etc.
To show how to use basic `ggplot2`, we'll use a dataset of Titanic passengers, their characteristics, and whether or not they survived the sinking. This dataset has become fairly famous in data science, because it's used, among other things, for one of Kaggle's long-term "learning" competitions, as well as in many tutorials and texts on building classification models.
Kaggle is a company that runs predictive modeling competitions, with top competitors sometimes winning cash prizes or interviews at top companies. At any time, Kaggle is typically is hosting several competitions, including some with no cash reward that are offered to help users get started with predictive modeling.

Doing Kaggle competitions is a great way to practice working with data and building models. Kaggle also has forums, repositories of code script for different competitions, and a blog that includes tips from past winners.

To get this dataset, you'll need to install and load the `titanic` package, and then you can load and rename the training datasets, which includes data on about two-thirds of the Titanic passengers:
```
# install.packages("titanic") # If you don't have the package installed
library(titanic)
data("titanic_train", package = "titanic")
titanic <- titanic_train
```
The other data example we'll use in this subsection is some data on players in the 2010 World Cup. This is available from the `faraway` package:
```
# install.packages("faraway") # If you don't have the package installed
library(faraway)
data("worldcup")
```

Unlike most data objects you'll work with, the data that comes with an R package will often have its own help file. You can access this using the `?` operator. For example, try running: `?worldcup`.
All of the plots we'll make in this subsection will use the `ggplot2` package (another member of the tidyverse!). If you don't already have that installed, you'll need to install it. You then need to load the package in your current session of R:

```
# install.packages("ggplot2")  ## Uncomment and run if you don't have `ggplot2`
installed
library(ggplot2)
```

The process of creating a plot using `ggplot2` follows conventions that are a bit different than most of the code you've seen so far in R (although it is somewhat similar to the idea of piping we introduced in an earlier course). The basic steps behind creating a plot with `ggplot2` are:

1. Create an object of the `ggplot` class, typically specifying the **data** and some or all of the **aesthetics**;
2. Add on **geoms** and other elements to create and customize the plot, using `+`.

You can add on one or many geoms and other elements to create plots that range from very simple to very customized. We'll focus on simple geoms and added elements first, and then explore more detailed customization later.

Initializing a `ggplot` object

The first step in creating a plot using `ggplot2` is to create a ggplot object. This object will not, by itself, create a plot with anything in it. Instead, it typically specifies the data frame you want to use and which aesthetics will be mapped to certain columns of that data frame (aesthetics are explained more in the next subsection).
Use the following conventions to initialize a ggplot object:

```
## Generic code
object <- ggplot(dataframe, aes(x = column_1, y = column_2))
## or, if you don't need to save the object
ggplot(dataframe, aes(x = column_1, y = column_2))
```

The dataframe is the first parameter in a `ggplot` call and, if you like, you can use the parameter definition with that call (e.g., `data = dataframe`). Aesthetics are defined within an `aes` function call that typically is used within the `ggplot` call.
In `ggplot2`, life is much easier if everything you want to plot is included in a dataframe as a column, and the first argument to `ggplot` must be a dataframe. This format has been a bit hard for some base R graphics users to adjust to, since base R graphics tends to plot based on vector, rather than dataframe, inputs. Trying to pass in a vector rather than a dataframe can be a common reason for `ggplot2` errors for all R users.

Plot aesthetics

Aesthetics are properties of the plot that can show certain elements of the data. For example, in Figure 4.1, color shows (i.e., is mapped to) gender, x-position shows height, and y-position shows weight in a sample data set of measurements of children in Nepal.

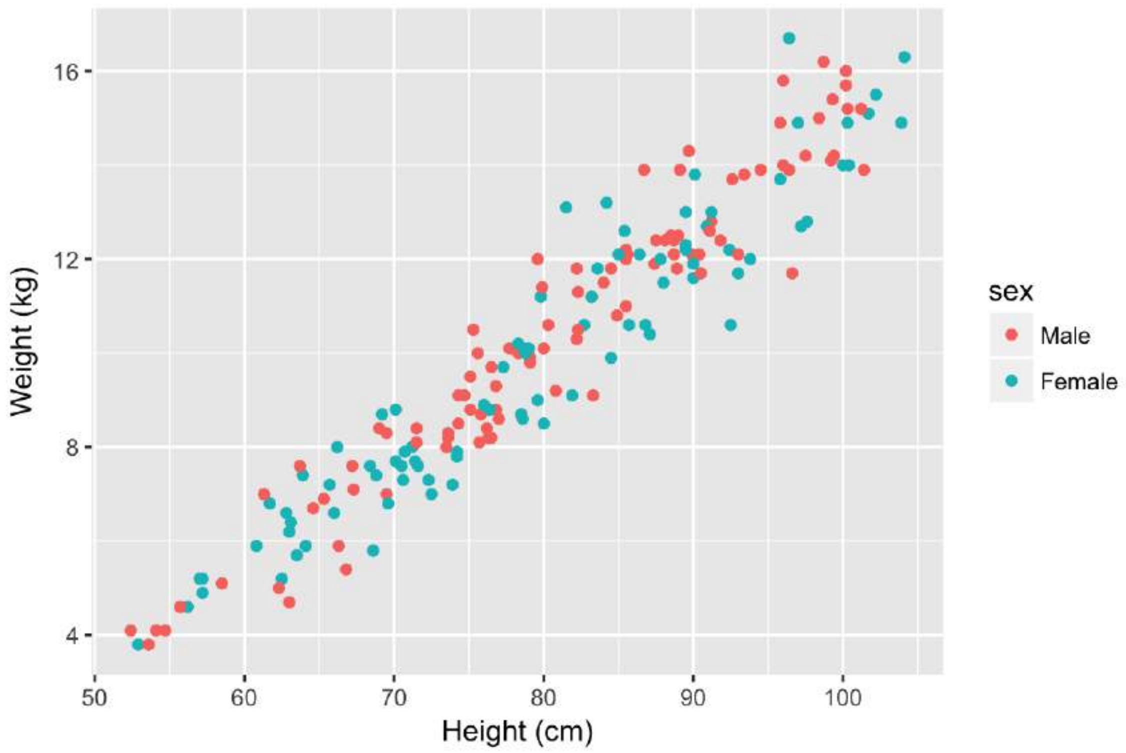


Figure 4.1: Example of how different properties of a plot can show different elements to the data. Here, color indicates gender, position along the x-axis shows height, and position along the y-axis shows weight. This example is a subset of data from the `nepali` dataset in the `faraway` package.

Any of these aesthetics could also be given a constant value, instead of being mapped to an element of the data. For example, all the points could be red, instead of showing gender. Later in this section, we will describe how to use these constant values for aesthetics. We'll discuss how to code this later in this section.

Which aesthetics are required for a plot depend on which geoms (more on those in a second) you're adding to the plot. You can find out the aesthetics you can use for a geom in the "Aesthetics" section of the geom's help file (e.g., `?geom_point`). Required aesthetics are in bold in this section of the help file and optional ones are not. Common plot aesthetics you might want to specify include:

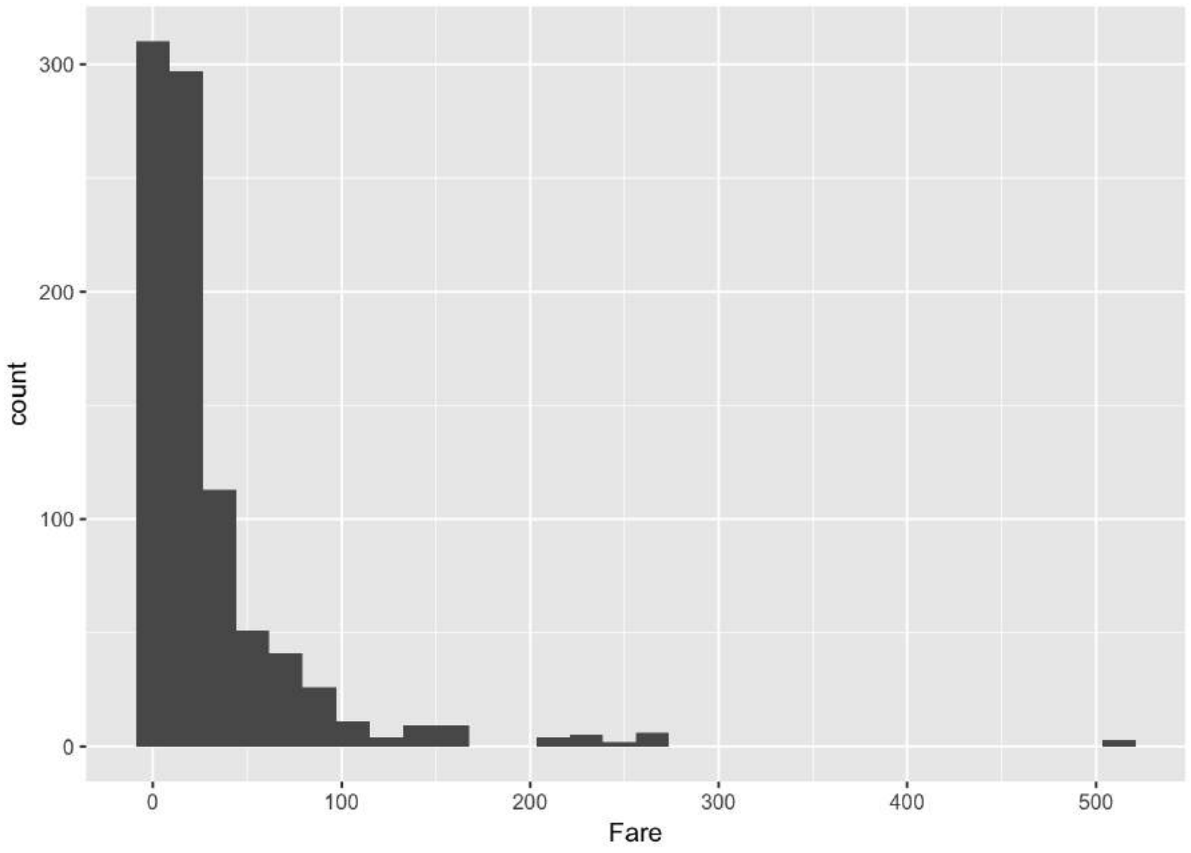
Code	**Description**
`x`	Position on x-axis
`y`	Position on y-axis
`shape`	Shape
`color`	Color of border of elements
`fill`	Color of inside of elements
`size`	Size
`alpha`	Transparency (1: opaque; 0: transparent)

`linetype`	Type of line (e.g., solid, dashed)

Creating a basic ggplot plot

To create a plot, you need to add one of more geoms to the ggplot object. The system of creating a `ggplot` object, mapping aesthetics to columns of the data, and adding geoms makes more sense once you try a few plots. For example, say you'd like to create a histogram showing the fares paid by passengers in the example Titanic data set. To plot the histogram, you'll first need to create a `ggplot` object, using a dataframe with the "Fares" column you want to show in the plot. In creating this `ggplot` object, you only need one aesthetic (`x`, which in this case you want to map to "Fares"), and then you'll need to add a histogram geom. In code, this is:

```
ggplot(data = titanic, aes(x = Fare)) +
  geom_histogram()
```



This code sets the dataframe as the `titanic` object in the user's working session, maps the values in the `Fare` column to the `x` aesthetic, and adds a histogram geom to generate a histogram.
If R gets to the end of a line and there is not some indication that the call is not over (e.g., `%>%` for piping or `+` for `ggplot2` plots), R interprets that as a message to run the call without reading in further code. A common error when writing `ggplot2` code is to put the `+` to add a geom or element at the beginning of a line rather than the end of a previous line— in this case, R will try to execute the call too soon. To avoid errors, be sure to end lines with `+`, don't start lines with it.
There is some flexibility in writing the code to create this plot. For example, you could specify the aesthetic for the histogram in an `aes` statement when adding the geom (`geom_histogram`) rather than in the `ggplot` call:

```
ggplot(data = titanic) +
  geom_histogram(aes(x = Fare))
```

Similarly, you could specify the dataframe when adding the geom rather than in the `ggplot` call:

```
ggplot() +
  geom_histogram(data = titanic, aes(x = Fare))
```

Finally, you can pipe the `titanic` dataframe into a `ggplot` call, since the `ggplot` function takes a dataframe as its first argument:

```r
titanic %>%
  ggplot() +
  geom_histogram(aes(x = Fare))
# or
titanic %>%
  ggplot(aes(x = Fare)) +
  geom_histogram()
```

While all of these work, for simplicity we will use the syntax of specifying the data and aesthetics in the `ggplot` call for most examples in this subsection. Later, we'll show how this flexibility can be used to use data from differents dataframe for different geoms or change aesthetic mappings between geoms.

A key thing to remember, however, is that `ggplot` is **not** flexible about whether you specify aesthetics within an `aes` call or not. We will discuss what happens if you do not later in the book, but it is very important that if you want to show values from a column of the data using aesthetics like color, size, shape, or position, you remember to make that specification within `aes`. Also, be sure that you specify the dataframe before or when you specify aesthetics (i.e., you can't specify aesthetics in the `ggplot` statement if you haven't specified the dataframe yet), and if you specify a dataframe within a geom, be sure to use `data =` syntax rather than relying on parameter position, as `data` is not the first parameter expected for geom functions.

When you run the code to create a plot in RStudio, the plot will be shown in the "Plots" tab in one of the RStudio panels. If you would like to save the plot, you can do so using the "Export" button in this tab. However, if you would like to use code in an R script to save a plot, you can do so (and it's more reproducible!).

To save a plot using code in a script, take the following steps: (1) open a graphics device (e.g., using the function `pdf` or `png`); (2) run the code to draw the map; and (3) close the graphics device using the `dev.off` function. Note that the function you use to open a graphics device will depend on the type of device you want to open, but you close all devices with the same function (`dev.off`).

Geoms

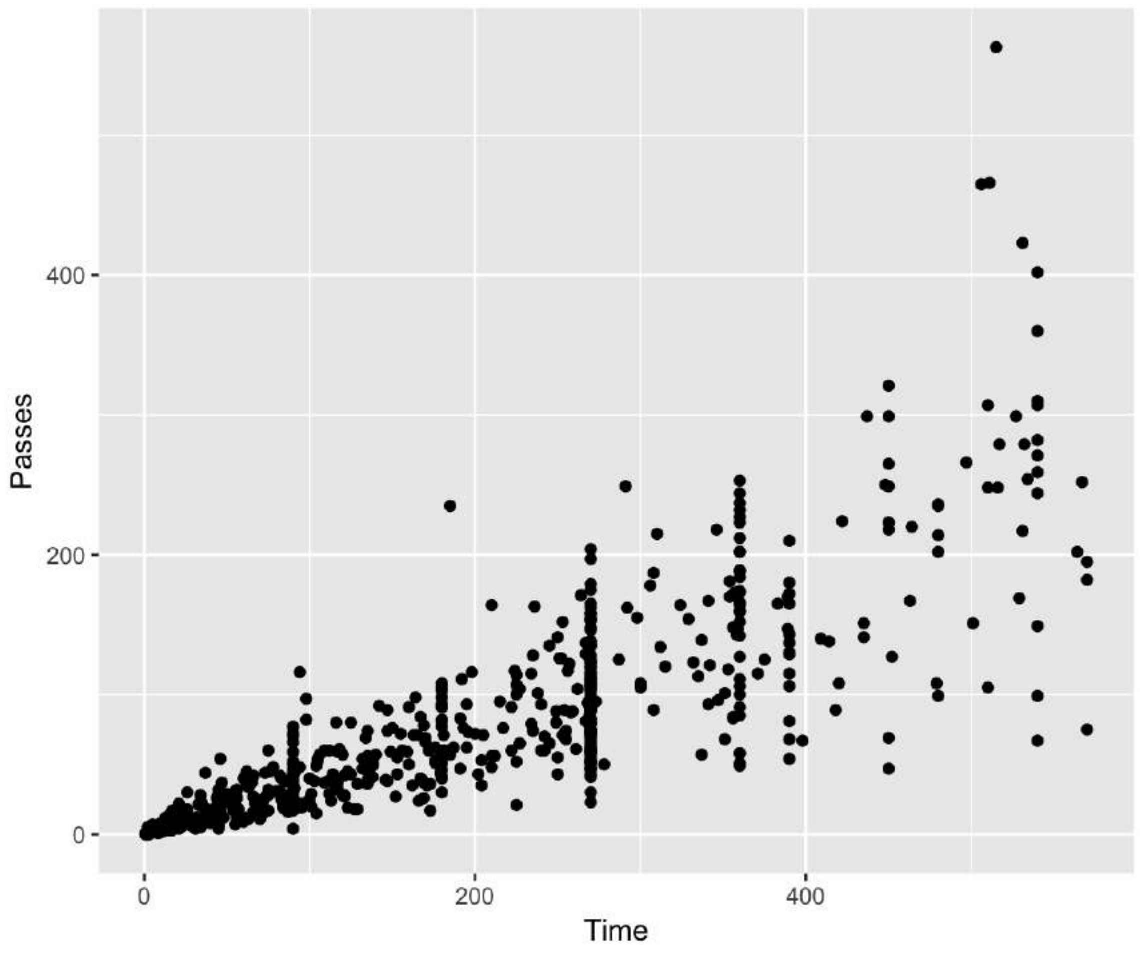
Geom functions add the graphical elements of the plot; if you do not include at least one geom, you'll get a blank plot space. Each geom function has its own arguments to adjust how the graph is created. For example, when adding a historgram geom, you can use the `bins` argument to change the number of bins used to create the histogram— try:

```r
ggplot(titanic, aes(x = Fare)) +
  geom_histogram(bins = 15)
```

As with any R functions, you can find out more about the arguments available for a geom function by reading the function's help file (e.g., `?geom_histogram`).

Geom functions differ in the aesthetic inputs they require. For example, the `geom_histogram` funciton only requires a single aesthetic (`x`). If you want to create a scatterplot, you'll need two aesthetics, `x` and `y`. In the `worldcup` dataset, the `Time` column gives the amount of time each player played in the World Cup 2010 and the `Passes` column gives the number of passes he made. To see the relationship between these two variables, you can create a ggplot object with the dataframe, mapping the x aesthetic to `Time` and the y aesthetic to `Passes`, and then adding a point geom:

```r
ggplot(worldcup, aes(x = Time, y = Passes)) +
  geom_point()
```

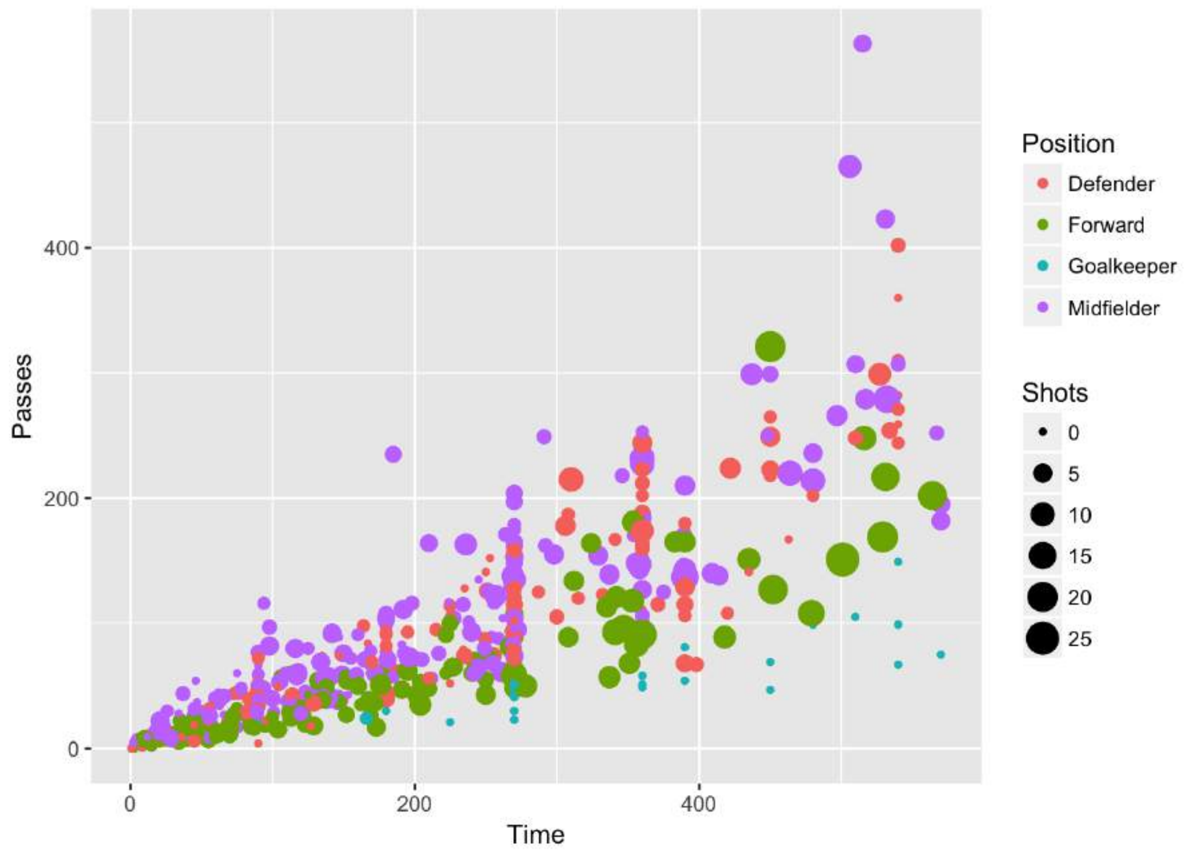


All geom functions have both *required* and *accepted* aesthetics. For example, the `geom_point` function requires `x` and `y`, but the function will also accept `alpha` (transparency), `color`, `fill`, `group`, `size`, `shape`, and `stroke` aesthetics. If you try to create a geom without one its required aesthetics, you will get an error:

```
ggplot(worldcup, aes(x = Time)) +
  geom_point()
Error: geom_point requires the following missing aesthetics: y
```

You can, however, add accepted aesthetics to the geom if you'd like; for example, to use color to show player position and size to show shots on goal for the World Cup data, you could call:

```
ggplot(worldcup, aes(x = Time, y = Passes,
                     color = Position, size = Shots)) +
  geom_point()
```



The following table gives some of the geom functions you may find useful in `ggplot2`, along with the required aesthetics and some of the most useful some useful specific arguments for each geom function (there are other useful arguments that can be applied to many different geom functions, which will be covered later). The elements created by these geom functions are usually clear from the function names (e.g., `geom_point` plots points; `geom_segment` plots segments).

Table 4.1: MVPs of geom functions

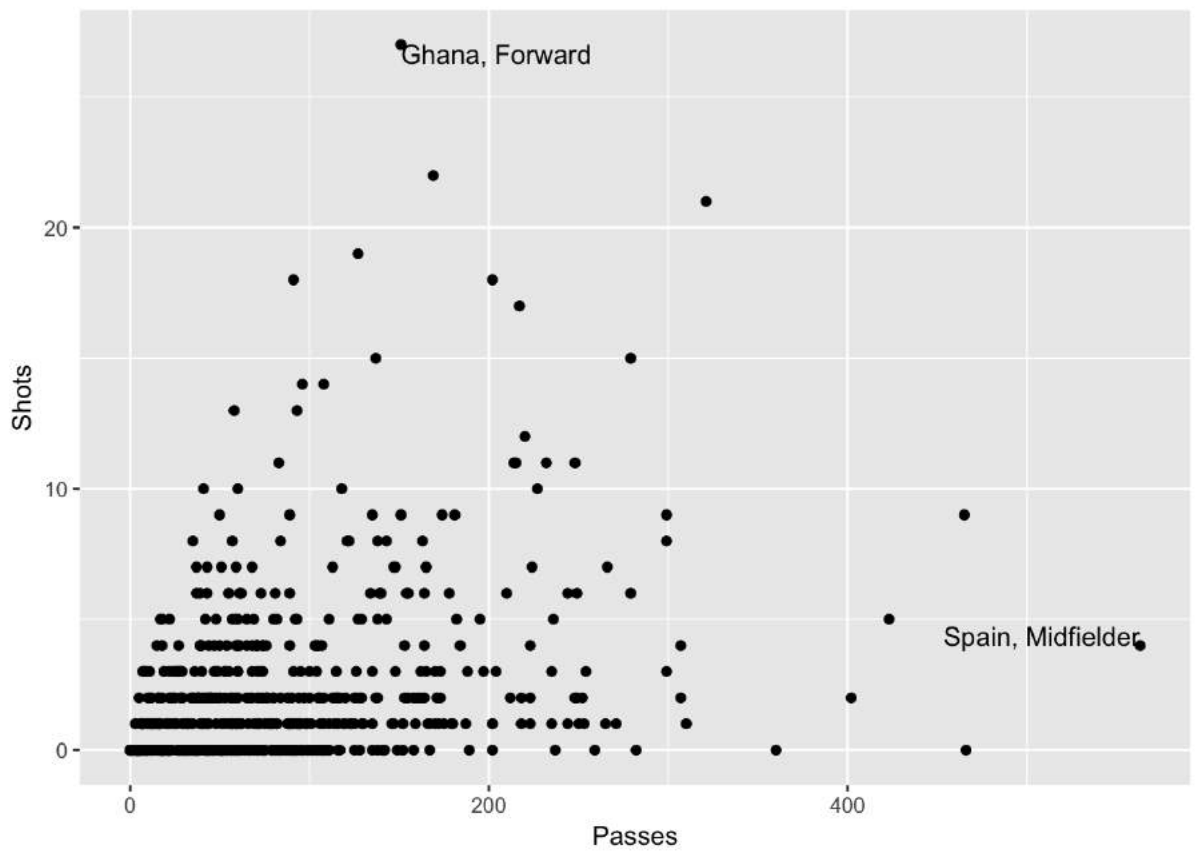
Function	**Common aesthetics**	**Common arguments**
`geom_point()`	`x`, `y`	
`geom_line()`	`x`, `y`	`arrow`, `na.rm`
`geom_segment()`	`x`, `y`, `xend`, `yend`	`arrow`, `na.rm`
`geom_path()`	`x`, `y`	`na.rm`
`geom_polygon()`	`x`, `y`	
`geom_histogram()`	`x`	`bins`, `binwidth`
`geom_abline()`	`intercept`, `slope`	

`geom_hline()`	`yintercept`	
`geom_vline()`	`xintercept`	
`geom_smooth()`	`x`, `y`	`method`, `se`, `span`
`geom_text()`	`x`, `y`, `label`	`parse`, `nudge_x`, `nudge_y`

Using multiple geoms

Several geoms can be added to the same `ggplot` object, which allows you to build up layers to create interesting graphs. For example, we previously made a scatterplot of time versus shots for World Cup 2010 data. You could make that plot more interesting by adding label points for noteworthy players with those players' team names and positions. First, you can create a subset of data with the information for noteworthy players and add a column with the text to include on the plot. Then you can add a text geom to the previous ggplot object:

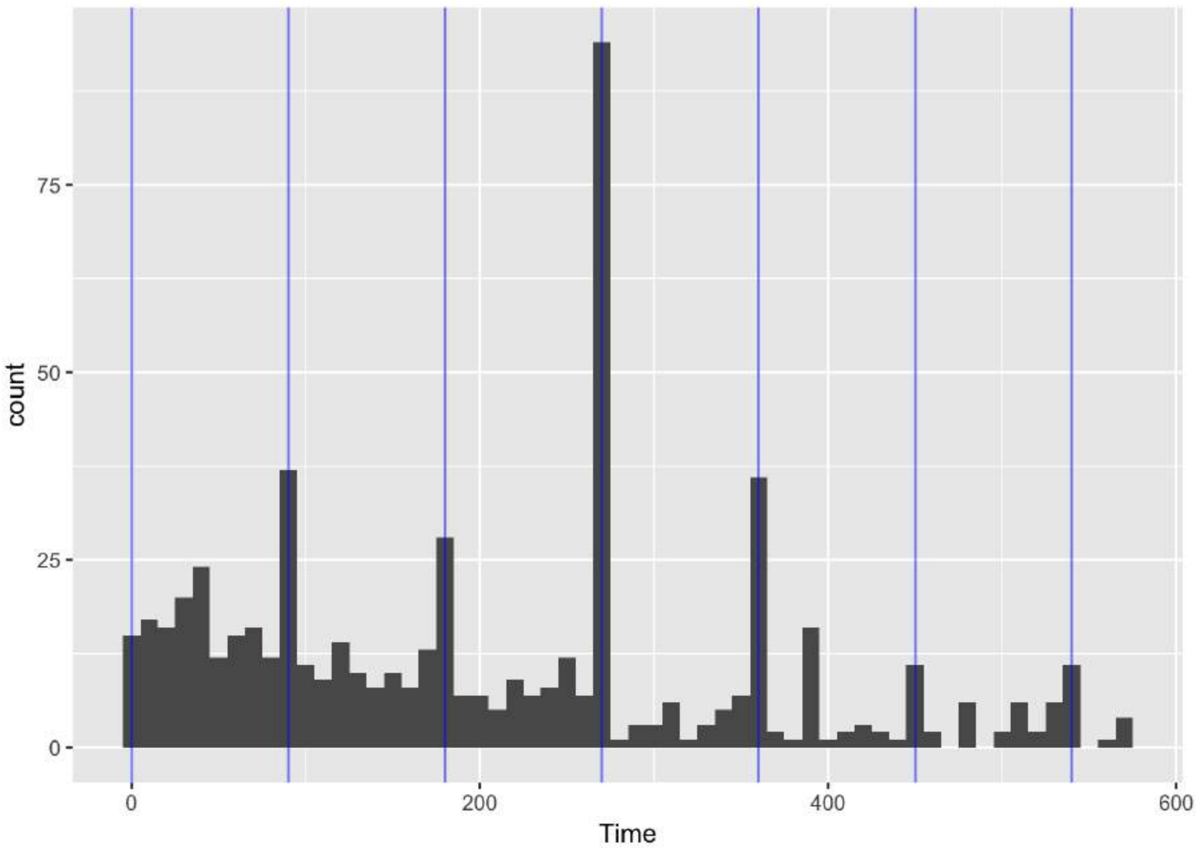
```r
library(dplyr)
noteworthy_players <- worldcup %>% filter(Shots == max(Shots) |
                                            Passes == max(Passes)) %>%
  mutate(point_label = paste(Team, Position, sep = ", "))

ggplot(worldcup, aes(x = Passes, y = Shots)) +
  geom_point() +
  geom_text(data = noteworthy_players, aes(label = point_label),
            vjust = "inward", hjust = "inward")
```



In this example, we're using data from different dataframes for different geoms. We'll discuss how that works more later in this section.

As another example, there seemed to be some horizontal clustering in the scatterplot we made of player time versus passes made for the `worldcup` data. Soccer games last 90 minutes each, and different teams play a different number of games at the World Cup, based on how well they do. To check if horizontal clustering is at 90-minute intervals, you can plot a histogram of player time (`Time`), with reference lines every 90 minutes. First initialize the ggplot object, with the dataframe to use and appropriate mapping to aesthetics, then add geoms for a histogram as well as vertical reference lines:

```
ggplot(worldcup, aes(x = Time)) +
        geom_histogram(binwidth = 10) +
        geom_vline(xintercept = 90 * 0:6,
                   color = "blue", alpha = 0.5)
```

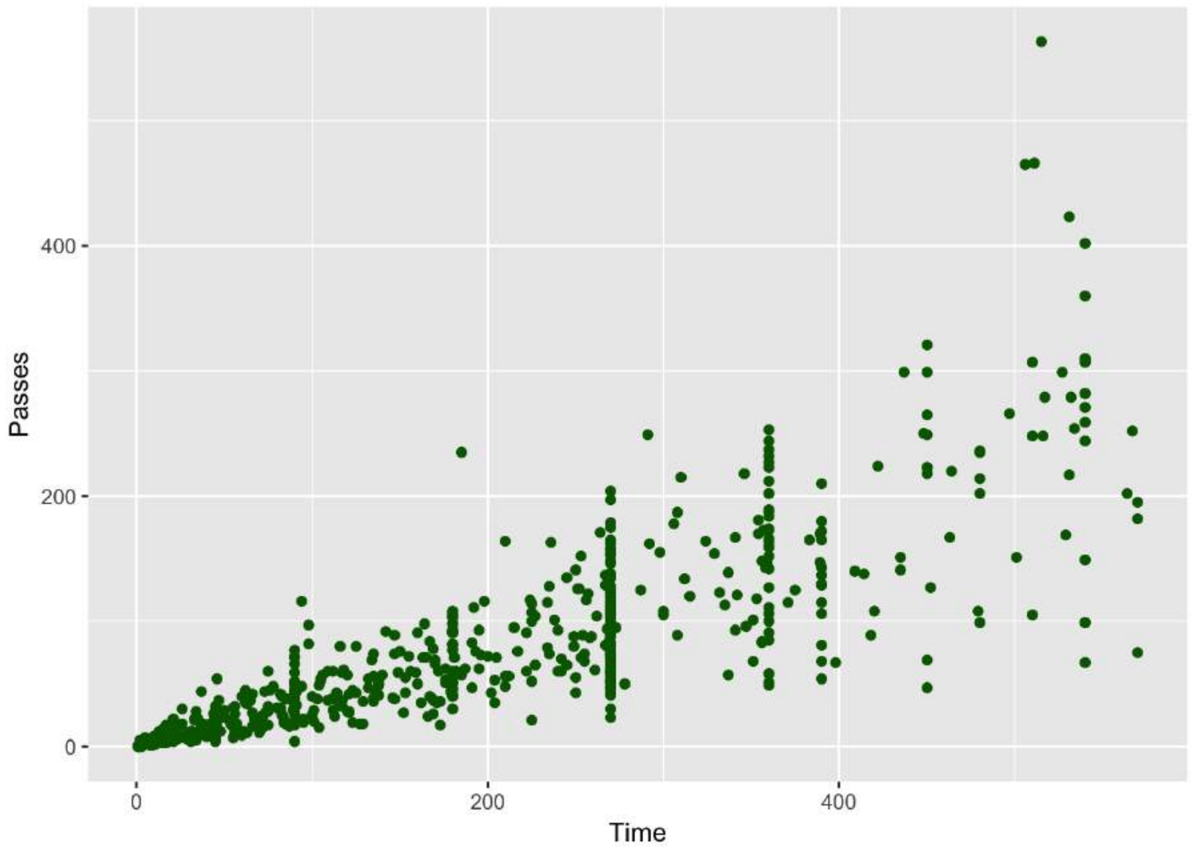


Based on this graph, player's times do cluster at 90-minute marks, especially at 270 minutes, which would be approximately after three games, the number played by all teams that fail to make it out of the group stage.

Constant aesthetics

Instead of mapping an aesthetic to an element of your data, you can use a constant value for it. For example, you may want to make all the points green in the World Cup scatterplot. You can do that by specifying the color aesthetic **outside** of an `aes` call when adding the points geom. For example:

```
ggplot(worldcup, aes(x = Time, y = Passes)) +
  geom_point(color = "darkgreen")
```



You can do this with any of the aesthetics for a geom, including color, fill, shape, and size. If you want to change the shape of points, in R, you use a number to specify the shape you want to use. Figure 4.2 shows the shapes that correspond to the numbers 1 to 25 in the `shape` aesthetic. This figure also provides an example of the difference between the *color* aesthetic (black for all these example points) and *fill* aesthetic (red for these examples). If a geom has both a border and an interior, the color aesthetic specifies the color of the border while the fill aesthetic specifies the color of the interior. You can see that, for point geoms, some shapes include a fill (21 for example), while some are either empty (1) or solid (19).

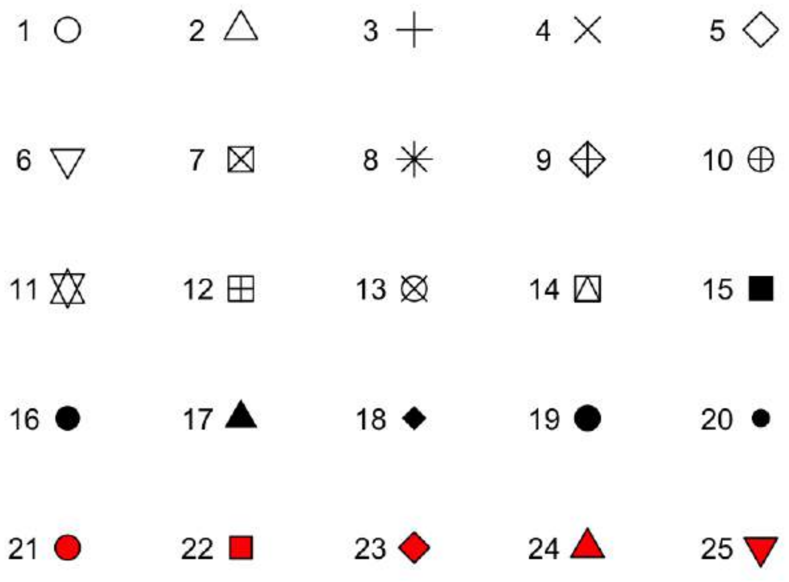


Figure 4.2: Examples of the shapes corresponding to different numeric choices for the `shape` aesthetic. For all examples, `color` is set to black and `fill` to red.

If you want to set color to be a constant value, you can do that in R using character strings for different colors. Figure 4.3 gives an example of a few of the different blues available in R. To find images that show all these named choices for colors in R, google "R colors" and search by "Images" (for example, there is a pdf here: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf).

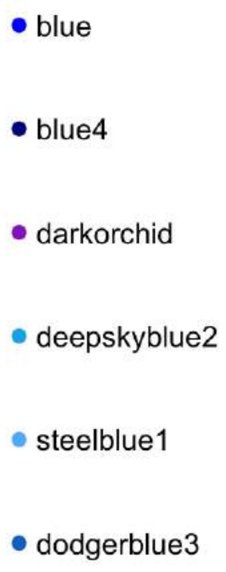


Figure 4.3: Example of a few available shades of blue in R.

Later we will cover additioal ways of handling colors in R, including different color palettes you can use. However, these "named" colors just shown can be a fast way to customize constant colors in R plots.

Other useful plot additions

There are also a number of elements besides geoms that you can add onto a `ggplot` object using `+`. A few that are used very frequently are:

Element	Description
`ggtitle`	Plot title
`xlab`, `ylab`	x- and y-axis labels
`xlim`, `ylim`	Limits of x- and y-axis

You can also use this syntax to customize plot scales and themes, which we will discuss later in this section.

Example plots

In this subsection, we'll show a few more examples of basic plots created with `ggplot2`. For the example plots in this subsection, we'll use a dataset in the `faraway` package called `nepali`. This gives data from a study of the health of a group of Nepalese children. You can load this data using:

```r
# install.packages("faraway") ## Uncomment if you do not have the faraway package
installed
library(faraway)
data(nepali)
```

Each observation in this dataframe represents a measurement for a child, including some physiological measurements like height and weight, and some children were measured multiple times and so have multiple observations in this data. Before plotting this data, we cleaned it a bit. We used tidyverse functions to select a subset of the columns: child id, sex, weight, height, and age. We also used the `distinct` function from `dplyr` to limit the dataset to the first measurement for each child.

```
nepali <- nepali %>%
  select(id, sex, wt, ht, age) %>%
  mutate(id = factor(id),
         sex = factor(sex, levels = c(1, 2),
                      labels = c("Male", "Female"))) %>%
  distinct(id, .keep_all = TRUE)
```

After this cleaning, the data looks like this:

```
head(nepali)
      id    sex   wt    ht age
1 120011   Male 12.8  91.2  41
2 120012 Female 14.9 103.9  57
3 120021 Female  7.7  70.1   8
4 120022 Female 12.1  86.4  35
5 120023   Male 14.2  99.4  49
6 120031   Male 13.9  96.4  46
```

We'll use this cleaned dataset to show how to use `ggplot2` to make histograms, scatterplots, and boxplots.

Histograms

Histograms show the distribution of a single variable. Therefore, `geom_histogram()` requires only one main aesthetic, `x`, which should be numeric. For example, to create a histogram of children's heights for the Nepali dataset (Figure 4.4), create a ggplot object with the data `nepali` and with the height column (`ht`) mapped to the ggplot object's x aesthetic. Then add a histogram geom:

```
ggplot(nepali, aes(x = ht)) +
  geom_histogram()
```

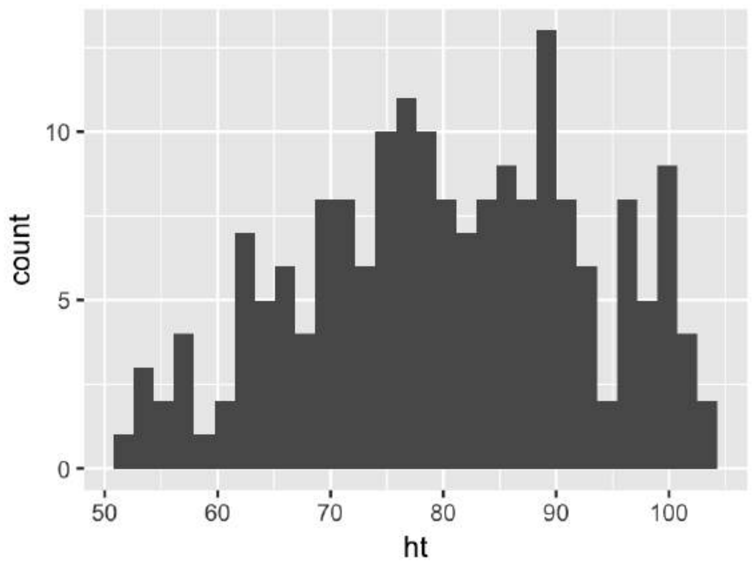


Figure 4.4: Basic example of plotting a histogram with `ggplot2`. This histogram shows the distribution of heights for the first recorded measurements of each child in the `nepali` dataset.

If you run the code with no arguments for `binwidth` or `bins` in `geom_histogram`, you will get a message saying "`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.". This message is just saying that a default number of bins was used to create the histogram. You can use arguments

to change the number of bins used, but often this default is fine. You may also get a message that observations with missing values were removed.
You can add some elements to this plot to customize it a bit. For example (Figure 4.5), you can add a figure title (`ggtitle`) and clearer labels for the x-axis (`xlab`). You can also change the range of values shown by the x-axis (`xlim`).

```
ggplot(nepali, aes(x = ht)) +
  geom_histogram(fill = "lightblue", color = "black") +
  ggtitle("Height of children") +
  xlab("Height (cm)") + xlim(c(0, 120))
```

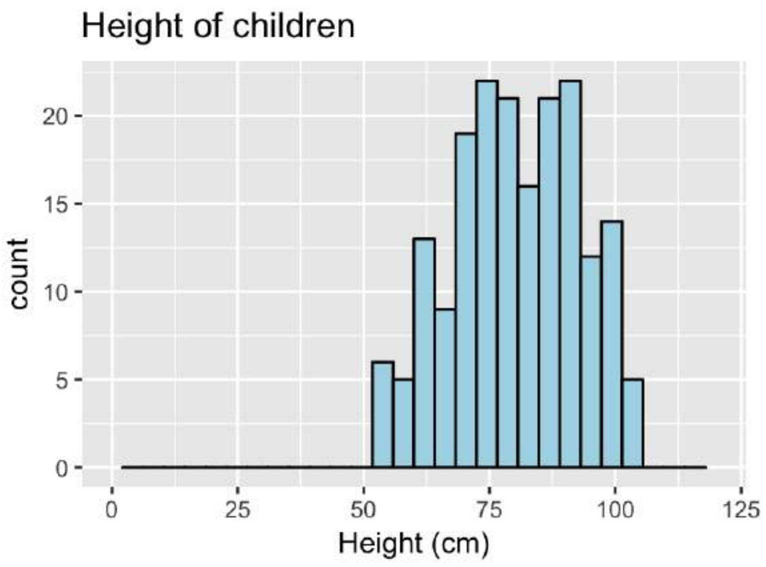


Figure 4.5: Example of adding ggplot elements to customize a histogram.

Note that these additional graphical elements are added on by adding function calls to `ggtitle`, `xlab`, and `xlim` to our ggplot object.

Scatterplots

A scatterplot shows the association between two variables. To create a scatterplot, add a point geom (`geom_point`) to a ggplot object. For example, to create a scatterplot of height versus age for the Nepali data (Figure 4.6), you can run the following code:

```
ggplot(nepali, aes(x = ht, y = wt)) +
  geom_point()
```

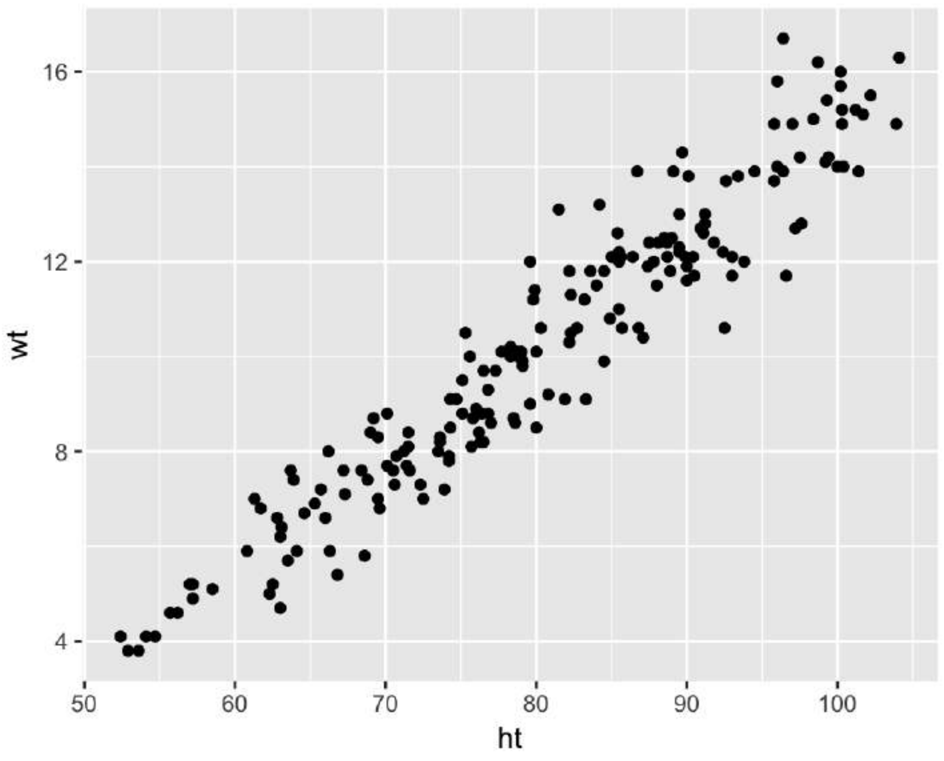


Figure 4.6: Example of creating a scatterplot. This scatterplot shows the relationship between children's heights and weights within the nepali dataset.

Again, you can use some of the options and additions to change the plot appearance. For example, to add a title, change the x- and y-axis labels, and change the color and size of the points on the scatterplot (Figure 4.7), you can run:

```
ggplot(nepali, aes(x = ht, y = wt)) +
  geom_point(color = "blue", size = 0.5) +
  ggtitle("Weight versus Height") +
  xlab("Height (cm)") + ylab("Weight (kg)")
```

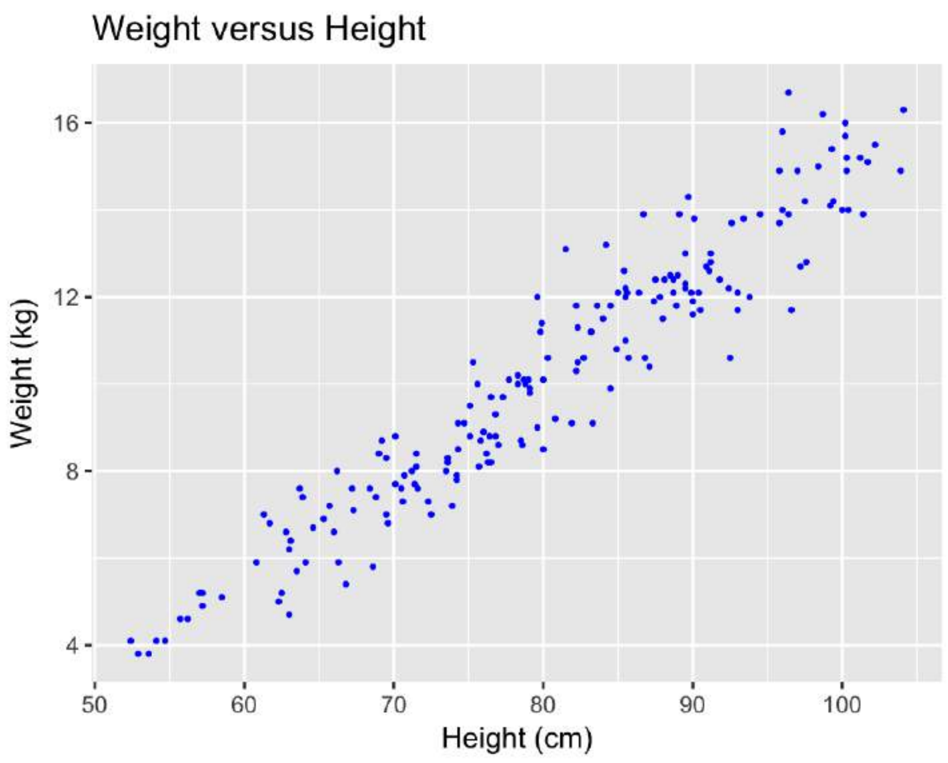


Figure 4.7: Example of adding ggplot elements to customize a scatterplot.

You can also try mapping a variable to the `color` aesthetic of the plot. For example, to use color to show the sex of each child in the scatterplot (Figure 4.8), you can run add an additional mapping of this optional aesthetic to the `sex` column of the `nepali` dataframe with the following code:

```
ggplot(nepali, aes(x = ht, y = wt, color = sex)) +
  geom_point(size = 0.5) +
  ggtitle("Weight versus Height") +
  xlab("Height (cm)") + ylab("Weight (kg)")
```

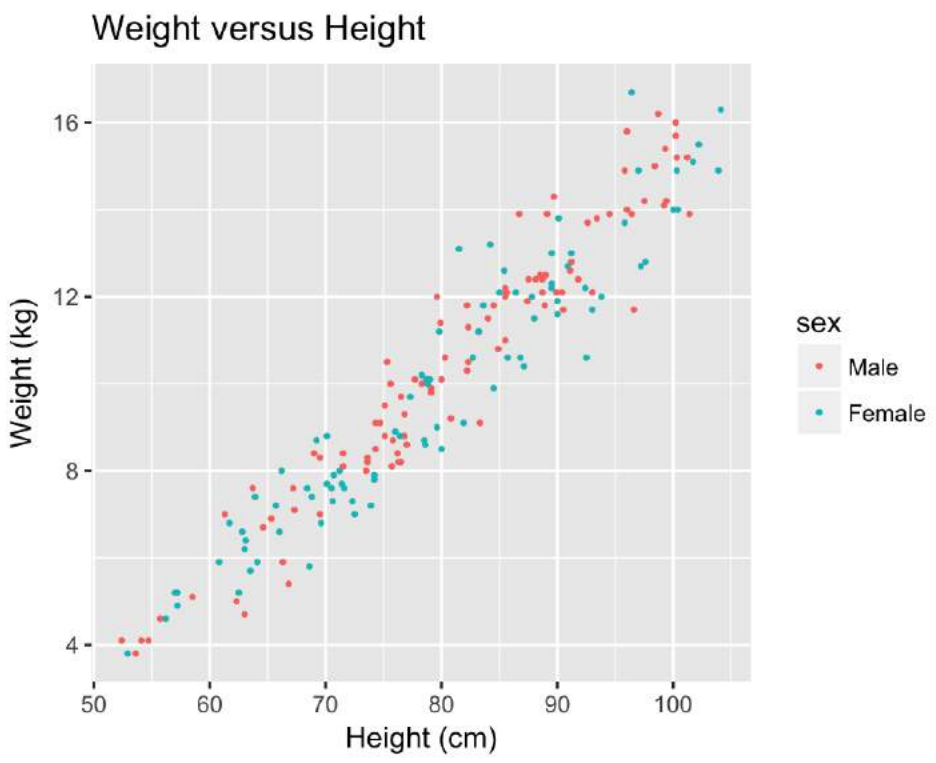


Figure 4.8: Example of mapping color to an element of the data in a scatterplot.

Boxplots

Boxplots are one way to show the distribution of a continuous variable. You can add a boxplot geom with the `geom_boxplot` function. To plot a boxplot for a single, continuous variable, you can map that variable to `y` in the `aes` call and map `x` to the constant `1`. For example, to create a boxplot of the heights of children in the Nepali dataset (Figure 4.9), you can run:

```r
ggplot(nepali, aes(x = 1, y = ht)) +
  geom_boxplot() +
  xlab("")+ ylab("Height (cm)")
```

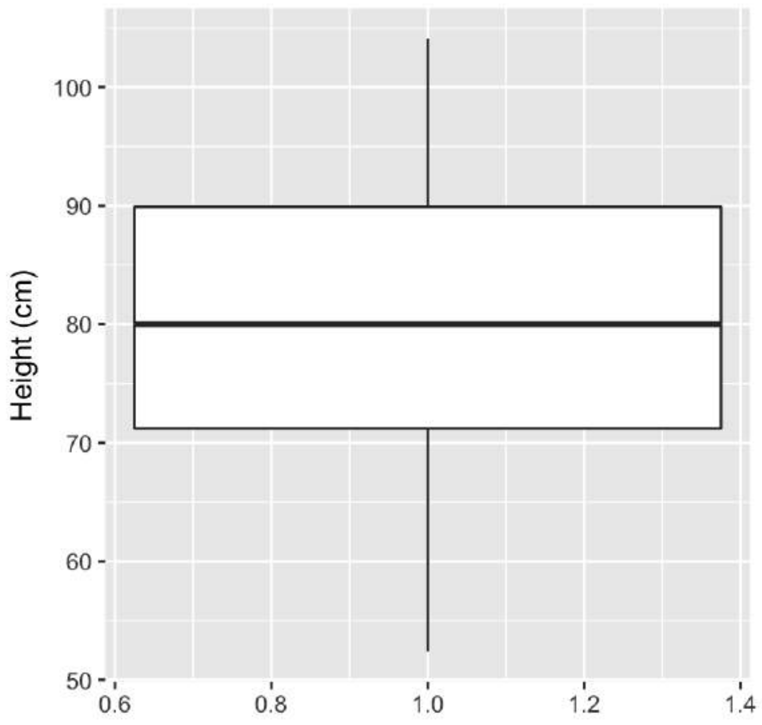


Figure 4.9: Example of creating a boxplot. The example shows the distribution of height data for children in the nepali dataset.

You can also create separate boxplots, one for each level of a factor (Figure 4.10). In this case, you'll need to map columns in the input dataframe to two aesthetics (x and y) when initializing the ggplot object The y variable is the variable for which the distribution will be shown, and the x variable should be a discrete (categorical or TRUE/FALSE) variable, which will be used to group the variable.

```
ggplot(nepali, aes(x = sex, y = ht)) +
  geom_boxplot() +
  xlab("Sex")+ ylab("Height (cm)")
```

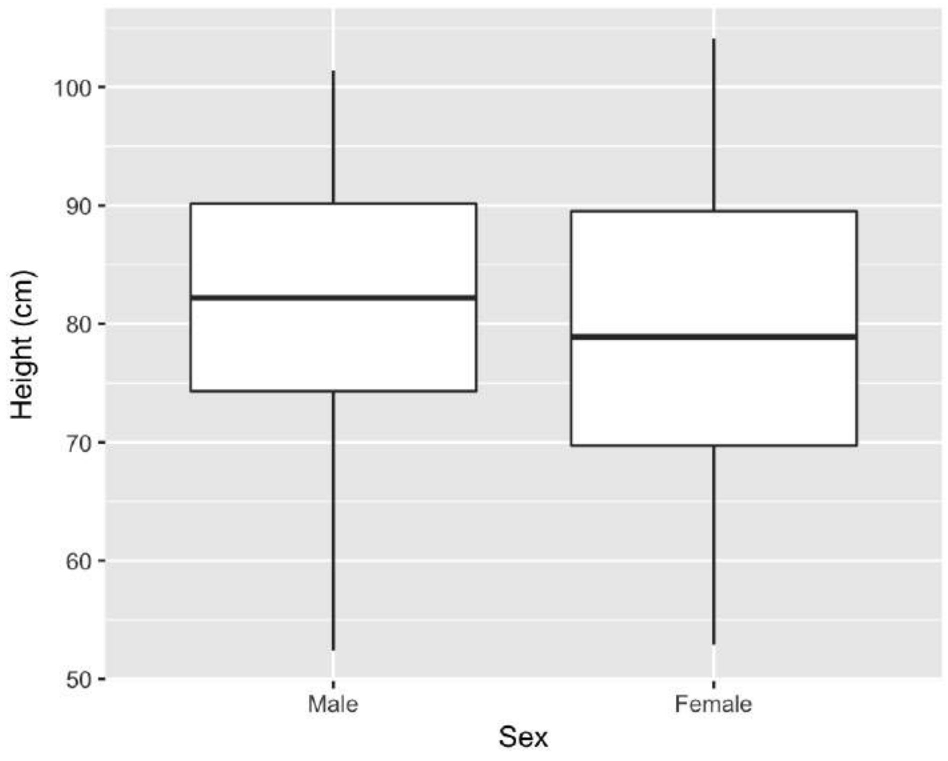


Figure 4.10: Example of creating separate boxplots, divided by a categorical grouping variable in the data.

Extensions of `ggplot2`

There are a number of packages that extend `ggplot2` and allow you to create a variety of interesting plots. For example, you can use the `ggpairs` function from the `GGally` package to plot all pairs of scatterplots for several variables (Figure 4.11).

```r
library(GGally)
ggpairs(nepali %>% select(sex, wt, ht, age))
```

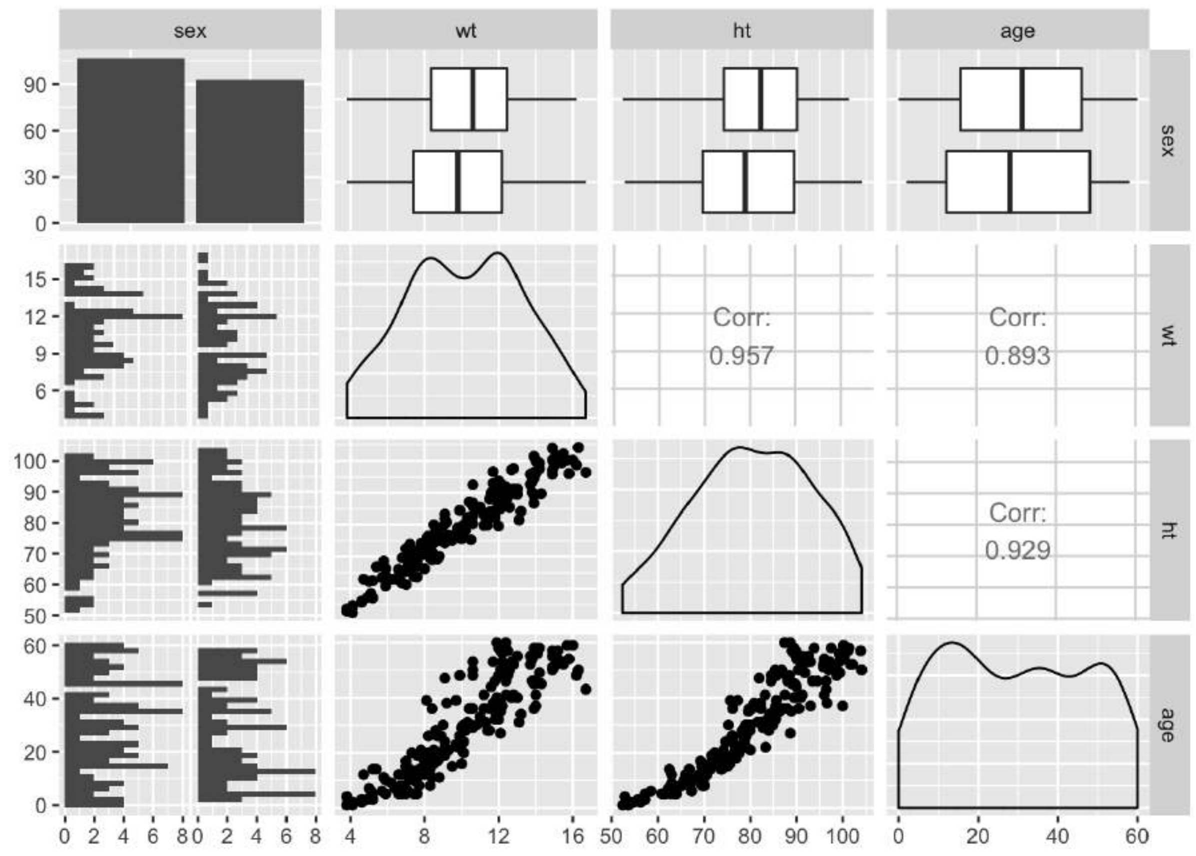


Figure 4.11: Example of using ggpairs from the GGally package for exploratory data analysis.

Notice how this output shows continuous and binary variables differently. For example, the center diagonal shows density plots for continuous variables, but a bar chart for the categorical variable.

See https://www.ggplot2-exts.org to find more `ggplot2` extensions. Later in this course, we will give an overview of how to make your own extensions.

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   ww. If you have any troubleshoot please contact your supervisor or Building Management
   xx. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   yy. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   zz. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 13

Date : December 10th, 2019
Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 9, students are expected to understand how to run an Application of data manipulation as it is about fitting models to data.

Procedure

Basic Plotting With ggplot2

The `ggplot2` package allows you to quickly plot attractive graphics and to visualize and explore data. Objects created with `ggplot2` can also be extensively customized with `ggplot2` functions (more on that in the next subsection), and because `ggplot2` is built using grid graphics, anything that cannot be customized using `ggplot2` functions can often be customized using grid graphics. While the structure of `ggplot2` code differs substantially from that of base R graphics, it offers a lot of power for the required effort. This first subsection focuses on **useful**, rather than **attractive** graphs, since this subsection focuses on exploring rather than presenting data. Later sections will give more information about making more attractive or customized plots, as you'd want to do for final reports, papers, etc.
To show how to use basic `ggplot2`, we'll use a dataset of Titanic passengers, their characteristics, and whether or not they survived the sinking. This dataset has become fairly famous in data science, because it's used, among other things, for one of Kaggle's long-term "learning" competitions, as well as in many tutorials and texts on building classification models.
Kaggle is a company that runs predictive modeling competitions, with top competitors sometimes winning cash prizes or interviews at top companies. At any time, Kaggle is typically is hosting several competitions, including some with no cash reward that are offered to help users get started with predictive modeling.

Doing Kaggle competitions is a great way to practice working with data and building models. Kaggle also has forums, repositories of code script for different competitions, and a blog that includes tips from past winners.

To get this dataset, you'll need to install and load the `titanic` package, and then you can load and rename the training datasets, which includes data on about two-thirds of the Titanic passengers:

```r
# install.packages("titanic") # If you don't have the package installed
library(titanic)
data("titanic_train", package = "titanic")
titanic <- titanic_train
```

The other data example we'll use in this subsection is some data on players in the 2010 World Cup. This is available from the `faraway` package:

```r
# install.packages("faraway") # If you don't have the package installed
library(faraway)
data("worldcup")
```

Unlike most data objects you'll work with, the data that comes with an R package will often have its own help file. You can access this using the `?` operator. For example, try running: `?worldcup`.

All of the plots we'll make in this subsection will use the `ggplot2` package (another member of the tidyverse!). If you don't already have that installed, you'll need to install it. You then need to load the package in your current session of R:

```r
# install.packages("ggplot2")  ## Uncomment and run if you don't have `ggplot2` installed
library(ggplot2)
```

The process of creating a plot using `ggplot2` follows conventions that are a bit different than most of the code you've seen so far in R (although it is somewhat similar to the idea of piping we introduced in an earlier course). The basic steps behind creating a plot with `ggplot2` are:

3. Create an object of the `ggplot` class, typically specifying the **data** and some or all of the **aesthetics**;
4. Add on **geoms** and other elements to create and customize the plot, using `+`.

You can add on one or many geoms and other elements to create plots that range from very simple to very customized. We'll focus on simple geoms and added elements first, and then explore more detailed customization later.

Initializing a `ggplot` object

The first step in creating a plot using `ggplot2` is to create a ggplot object. This object will not, by itself, create a plot with anything in it. Instead, it typically specifies the data frame you want to use and which aesthetics will be mapped to certain columns of that data frame (aesthetics are explained more in the next subsection).

Use the following conventions to initialize a ggplot object:

```r
## Generic code
object <- ggplot(dataframe, aes(x = column_1, y = column_2))
## or, if you don't need to save the object
ggplot(dataframe, aes(x = column_1, y = column_2))
```

The dataframe is the first parameter in a `ggplot` call and, if you like, you can use the parameter definition with that call (e.g., `data = dataframe`). Aesthetics are defined within an `aes` function call that typically is used within the `ggplot` call.

In `ggplot2`, life is much easier if everything you want to plot is included in a dataframe as a column, and the first argument to `ggplot` must be a dataframe. This format has been a bit hard for some base R graphics users to adjust to, since base R graphics tends to plot based on vector, rather than dataframe, inputs. Trying to pass in a vector rather than a dataframe can be a common reason for `ggplot2` errors for all R users.

Plot aesthetics

Aesthetics are properties of the plot that can show certain elements of the data. For example, in Figure 4.1, color shows (i.e., is mapped to) gender, x-position shows height, and y-position shows weight in a sample data set of measurements of children in Nepal.

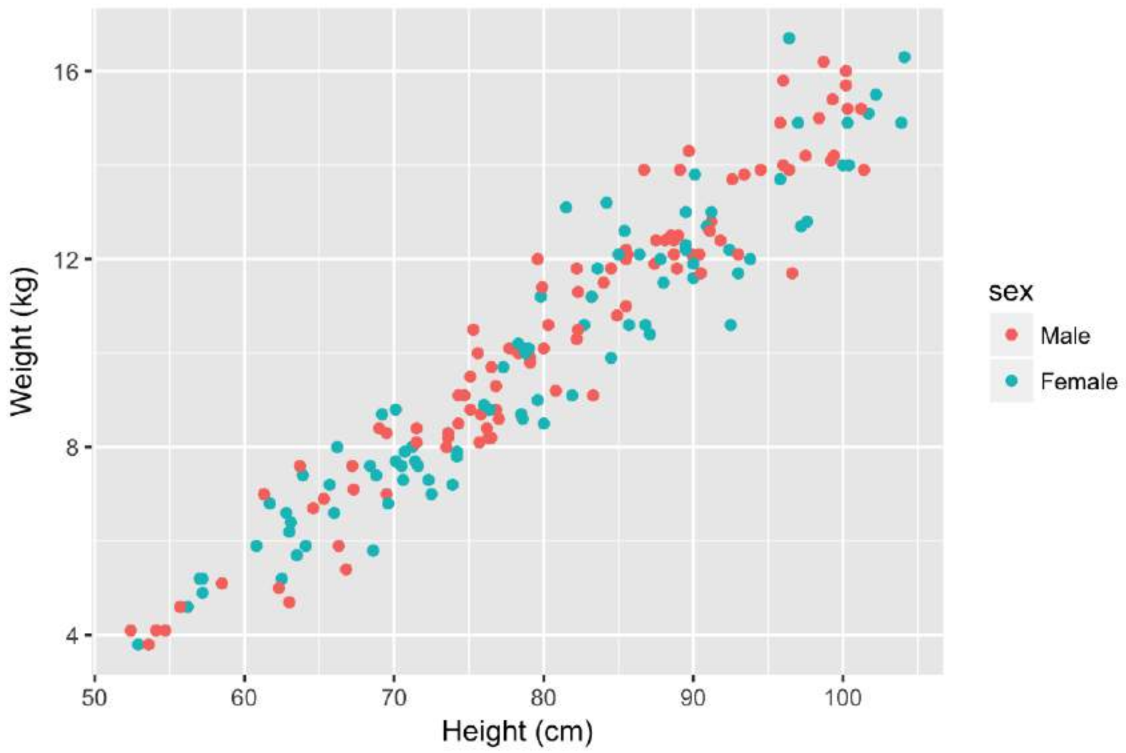


Figure 4.1: Example of how different properties of a plot can show different elements to the data. Here, color indicates gender, position along the x-axis shows height, and position along the y-axis shows weight. This example is a subset of data from the `nepali` dataset in the `faraway` package.
Any of these aesthetics could also be given a constant value, instead of being mapped to an element of the data. For example, all the points could be red, instead of showing gender. Later in this section, we will describe how to use these constant values for aesthetics. We'll discuss how to code this later in this section.

Which aesthetics are required for a plot depend on which geoms (more on those in a second) you're adding to the plot. You can find out the aesthetics you can use for a geom in the "Aesthetics" section of the geom's help file (e.g., `?geom_point`). Required aesthetics are in bold in this section of the help file and optional ones are not. Common plot aesthetics you might want to specify include:

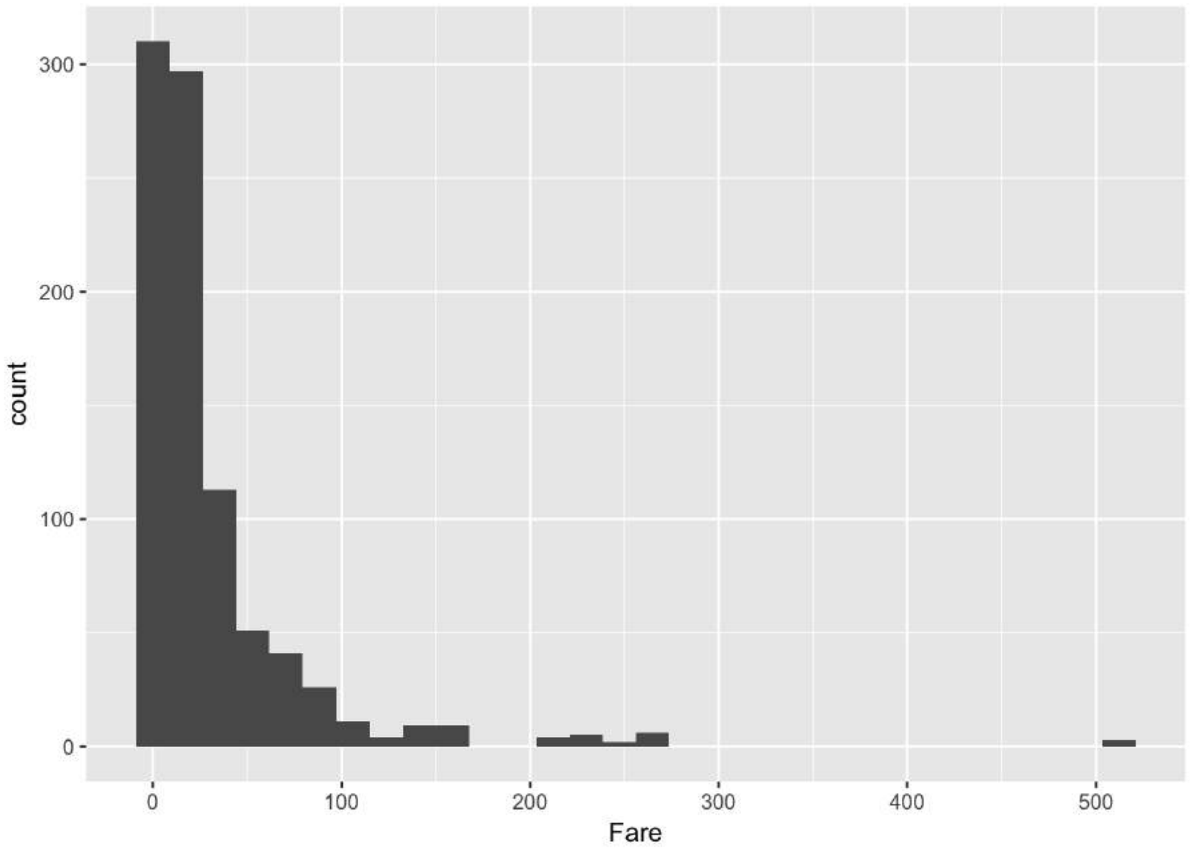
Code	**Description**
X	Position on x-axis
Y	Position on y-axis
`Shape`	Shape
`Color`	Color of border of elements
`Fill`	Color of inside of elements
`Size`	Size
`Alpha`	Transparency (1: opaque; 0: transparent)

`Linetype`	Type of line (e.g., solid, dashed)

Creating a basic ggplot plot

To create a plot, you need to add one of more geoms to the ggplot object. The system of creating a `ggplot` object, mapping aesthetics to columns of the data, and adding geoms makes more sense once you try a few plots. For example, say you'd like to create a histogram showing the fares paid by passengers in the example Titanic data set. To plot the histogram, you'll first need to create a `ggplot` object, using a dataframe with the "Fares" column you want to show in the plot. In creating this `ggplot` object, you only need one aesthetic (`x`, which in this case you want to map to "Fares"), and then you'll need to add a histogram geom. In code, this is:

```r
ggplot(data = titanic, aes(x = Fare)) +
  geom_histogram()
```



This code sets the dataframe as the `titanic` object in the user's working session, maps the values in the `Fare` column to the `x` aesthetic, and adds a histogram geom to generate a histogram.
If R gets to the end of a line and there is not some indication that the call is not over (e.g., `%>%` for piping or `+` for `ggplot2` plots), R interprets that as a message to run the call without reading in further code. A common error when writing `ggplot2` code is to put the `+` to add a geom or element at the beginning of a line rather than the end of a previous line— in this case, R will try to execute the call too soon. To avoid errors, be sure to end lines with `+`, don't start lines with it.
There is some flexibility in writing the code to create this plot. For example, you could specify the aesthetic for the histogram in an `aes` statement when adding the geom (`geom_histogram`) rather than in the `ggplot` call:

```r
ggplot(data = titanic) +
  geom_histogram(aes(x = Fare))
```

Similarly, you could specify the dataframe when adding the geom rather than in the `ggplot` call:

```r
ggplot() +
  geom_histogram(data = titanic, aes(x = Fare))
```

Finally, you can pipe the `titanic` dataframe into a `ggplot` call, since the `ggplot` function takes a dataframe as its first argument:

```
titanic %>%
  ggplot() +
  geom_histogram(aes(x = Fare))
# or
titanic %>%
  ggplot(aes(x = Fare)) +
  geom_histogram()
```

While all of these work, for simplicity we will use the syntax of specifying the data and aesthetics in the `ggplot` call for most examples in this subsection. Later, we'll show how this flexibility can be used to use data from differents dataframe for different geoms or change aesthetic mappings between geoms.

A key thing to remember, however, is that `ggplot` is **not** flexible about whether you specify aesthetics within an `aes` call or not. We will discuss what happens if you do not later in the book, but it is very important that if you want to show values from a column of the data using aesthetics like color, size, shape, or position, you remember to make that specification within `aes`. Also, be sure that you specify the dataframe before or when you specify aesthetics (i.e., you can't specify aesthetics in the `ggplot` statement if you haven't specified the dataframe yet), and if you specify a dataframe within a geom, be sure to use `data =` syntax rather than relying on parameter position, as `data` is not the first parameter expected for geom functions.

When you run the code to create a plot in RStudio, the plot will be shown in the "Plots" tab in one of the RStudio panels. If you would like to save the plot, you can do so using the "Export" button in this tab. However, if you would like to use code in an R script to save a plot, you can do so (and it's more reproducible!).

To save a plot using code in a script, take the following steps: (1) open a graphics device (e.g., using the function `pdf` or `png`); (2) run the code to draw the map; and (3) close the graphics device using the `dev.off` function. Note that the function you use to open a graphics device will depend on the type of device you want to open, but you close all devices with the same function (`dev.off`).

Geoms

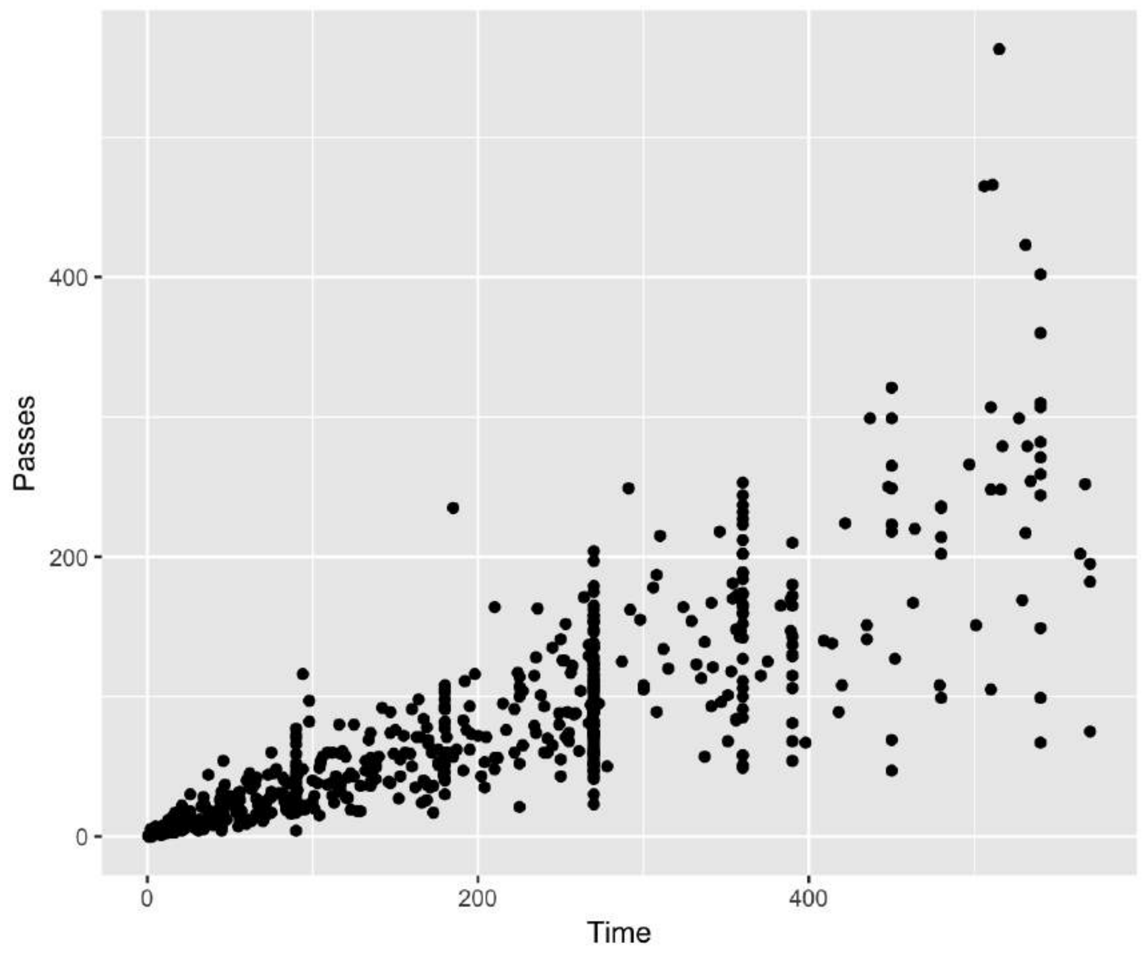
Geom functions add the graphical elements of the plot; if you do not include at least one geom, you'll get a blank plot space. Each geom function has its own arguments to adjust how the graph is created. For example, when adding a historgram geom, you can use the `bins` argument to change the number of bins used to create the histogram— try:

```
ggplot(titanic, aes(x = Fare)) +
  geom_histogram(bins = 15)
```

As with any R functions, you can find out more about the arguments available for a geom function by reading the function's help file (e.g., `?geom_histogram`).

Geom functions differ in the aesthetic inputs they require. For example, the `geom_histogram` funciton only requires a single aesthetic (`x`). If you want to create a scatterplot, you'll need two aesthetics, `x` and `y`. In the `worldcup` dataset, the `Time` column gives the amount of time each player played in the World Cup 2010 and the `Passes` column gives the number of passes he made. To see the relationship between these two variables, you can create a ggplot object with the dataframe, mapping the x aesthetic to `Time` and the y aesthetic to `Passes`, and then adding a point geom:

```
ggplot(worldcup, aes(x = Time, y = Passes)) +
  geom_point()
```

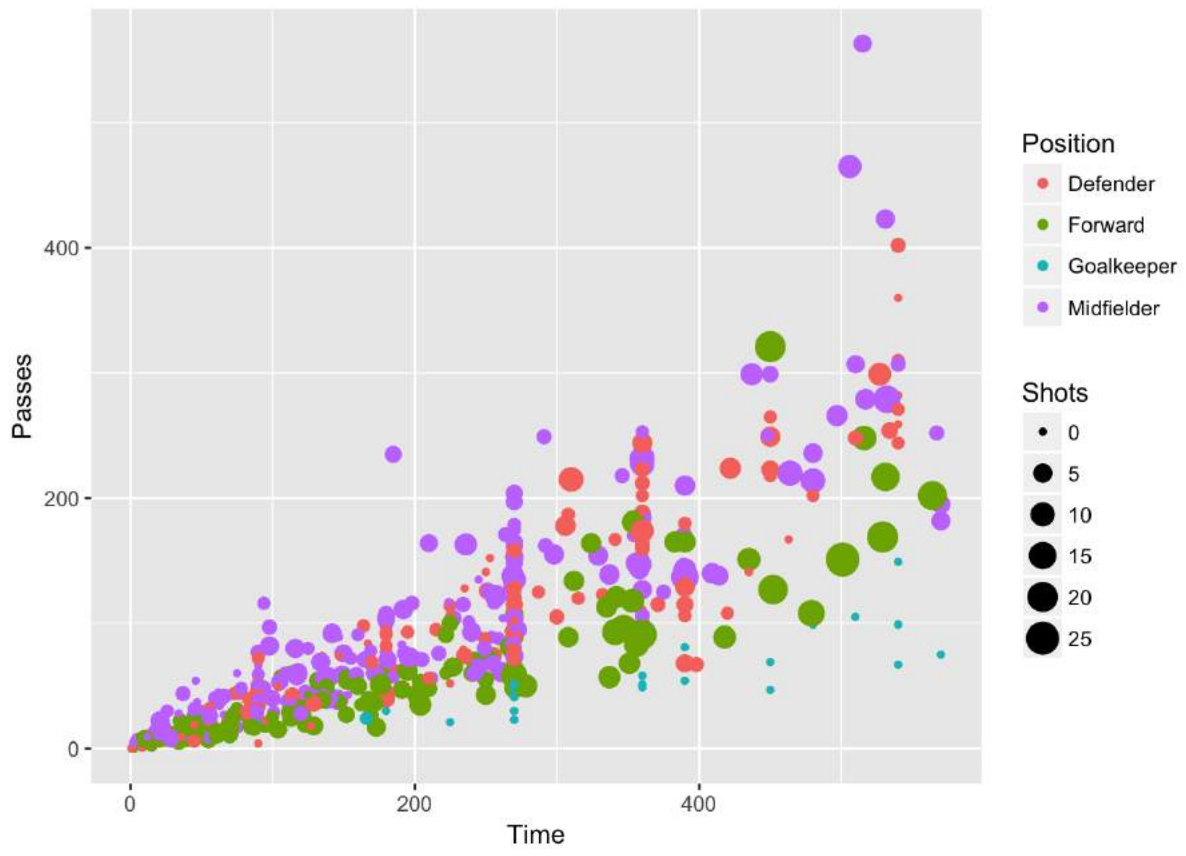


All geom functions have both *required* and *accepted* aesthetics. For example, the `geom_point` function requires `x` and `y`, but the function will also accept `alpha` (transparency), `color`, `fill`, `group`, `size`, `shape`, and `stroke` aesthetics. If you try to create a geom without one its required aesthetics, you will get an error:

```
ggplot(worldcup, aes(x = Time)) +
  geom_point()
Error: geom_point requires the following missing aesthetics: y
```

You can, however, add accepted aesthetics to the geom if you'd like; for example, to use color to show player position and size to show shots on goal for the World Cup data, you could call:

```
ggplot(worldcup, aes(x = Time, y = Passes,
                     color = Position, size = Shots)) +
  geom_point()
```



The following table gives some of the geom functions you may find useful in `ggplot2`, along with the required aesthetics and some of the most useful some useful specific arguments for each geom function (there are other useful arguments that can be applied to many different geom functions, which will be covered later). The elements created by these geom functions are usually clear from the function names (e.g., `geom_point` plots points; `geom_segment` plots segments).

Table 4.1: MVPs of geom functions

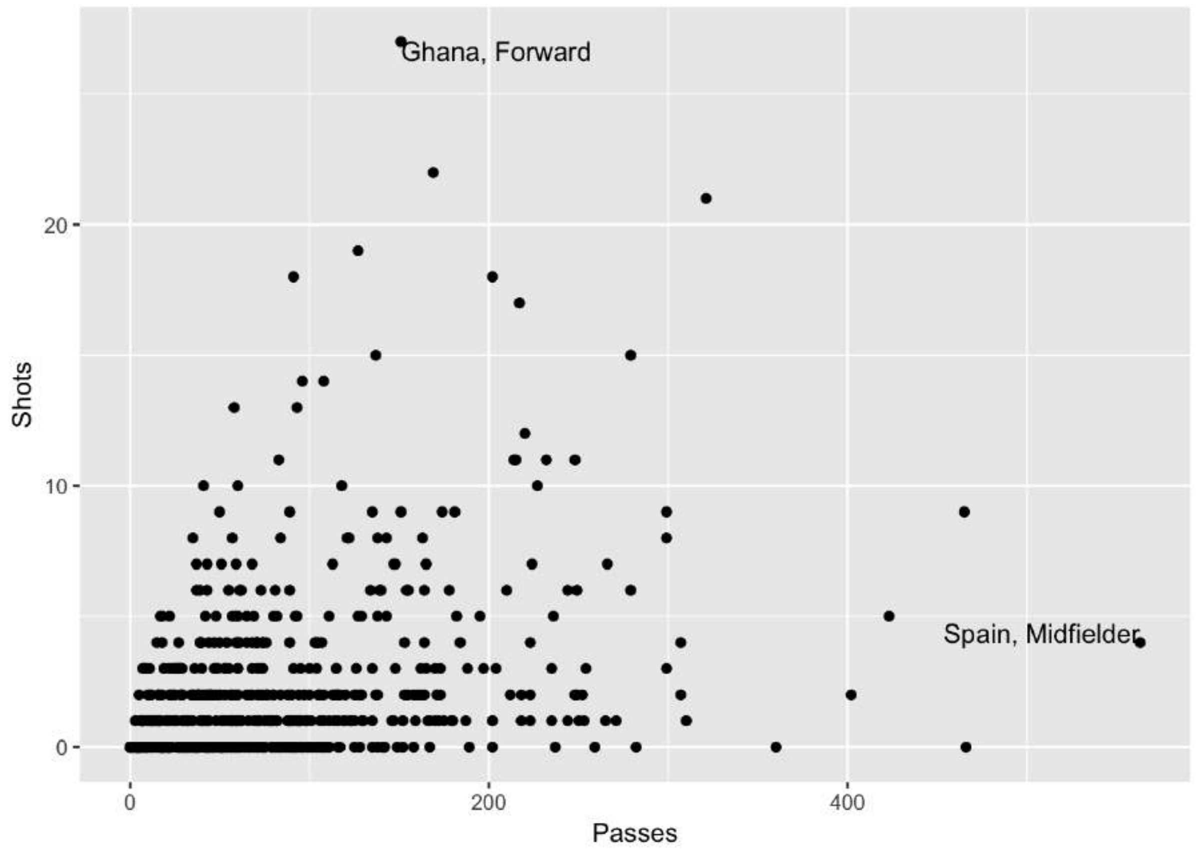
Function	**Common aesthetics**	**Common arguments**
`geom_point()`	`x`, `y`	
`geom_line()`	`x`, `y`	`arrow`, `na.rm`
`geom_segment()`	`x`, `y`, `xend`, `yend`	`arrow`, `na.rm`
`geom_path()`	`x`, `y`	`na.rm`
`geom_polygon()`	`x`, `y`	
`geom_histogram()`	`x`	`bins`, `binwidth`
`geom_abline()`	`intercept`, `slope`	

`geom_hline()`	`yintercept`	
`geom_vline()`	`xintercept`	
`geom_smooth()`	`x`, `y`	`method`, `se`, `span`
`geom_text()`	`x`, `y`, `label`	`parse`, `nudge_x`, `nudge_y`

Using multiple geoms

Several geoms can be added to the same `ggplot` object, which allows you to build up layers to create interesting graphs. For example, we previously made a scatterplot of time versus shots for World Cup 2010 data. You could make that plot more interesting by adding label points for noteworthy players with those players' team names and positions. First, you can create a subset of data with the information for noteworthy players and add a column with the text to include on the plot. Then you can add a text geom to the previous ggplot object:

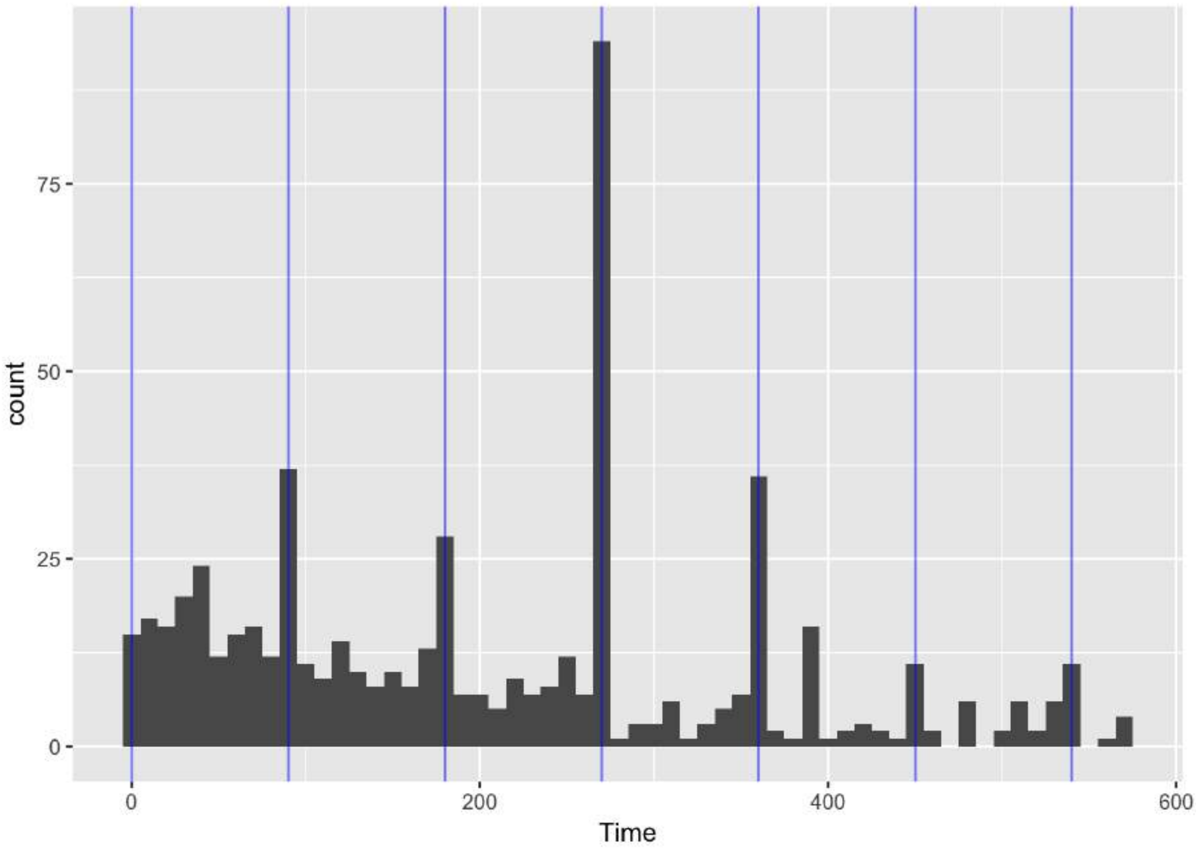
```
library(dplyr)
noteworthy_players <- worldcup %>% filter(Shots == max(Shots) |
                                           Passes == max(Passes)) %>%
  mutate(point_label = paste(Team, Position, sep = ", "))

ggplot(worldcup, aes(x = Passes, y = Shots)) +
  geom_point() +
  geom_text(data = noteworthy_players, aes(label = point_label),
            vjust = "inward", hjust = "inward")
```



In this example, we're using data from different dataframes for different geoms. We'll discuss how that works more later in this section.

As another example, there seemed to be some horizontal clustering in the scatterplot we made of player time versus passes made for the `worldcup` data. Soccer games last 90 minutes each, and different teams play a different number of games at the World Cup, based on how well they do. To check if horizontal clustering is at 90-minute intervals, you can plot a histogram of player time (`Time`), with reference lines every 90 minutes. First initialize the ggplot object, with the dataframe to use and appropriate mapping to aesthetics, then add geoms for a histogram as well as vertical reference lines:

```
ggplot(worldcup, aes(x = Time)) +
        geom_histogram(binwidth = 10) +
        geom_vline(xintercept = 90 * 0:6,
                   color = "blue", alpha = 0.5)
```

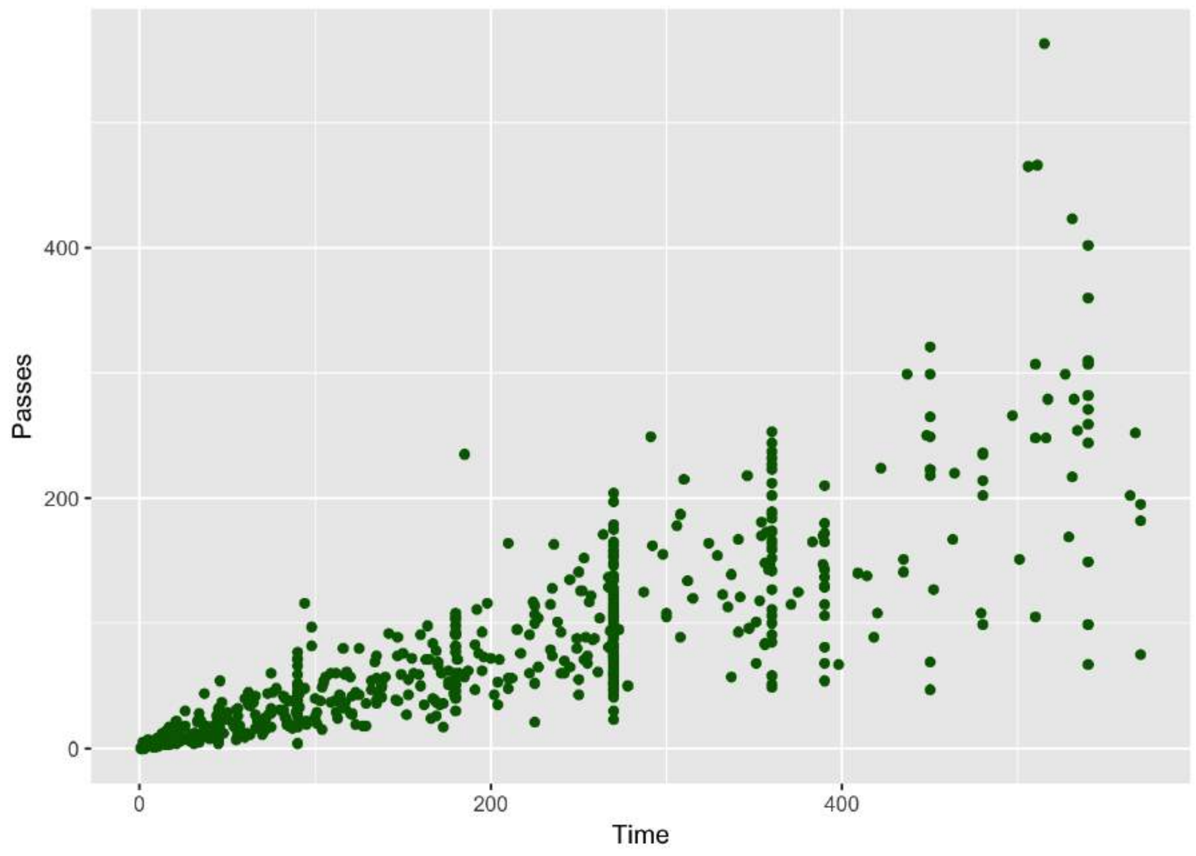


Based on this graph, player's times do cluster at 90-minute marks, especially at 270 minutes, which would be approximately after three games, the number played by all teams that fail to make it out of the group stage.

Constant aesthetics

Instead of mapping an aesthetic to an element of your data, you can use a constant value for it. For example, you may want to make all the points green in the World Cup scatterplot. You can do that by specifying the color aesthetic **outside** of an `aes` call when adding the points geom. For example:

```
ggplot(worldcup, aes(x = Time, y = Passes)) +
  geom_point(color = "darkgreen")
```



You can do this with any of the aesthetics for a geom, including color, fill, shape, and size. If you want to change the shape of points, in R, you use a number to specify the shape you want to use. Figure 4.2 shows the shapes that correspond to the numbers 1 to 25 in the `shape` aesthetic. This figure also provides an example of the difference between the *color* aesthetic (black for all these example points) and *fill* aesthetic (red for these examples). If a geom has both a border and an interior, the color aesthetic specifies the color of the border while the fill aesthetic specifies the color of the interior. You can see that, for point geoms, some shapes include a fill (21 for example), while some are either empty (1) or solid (19).

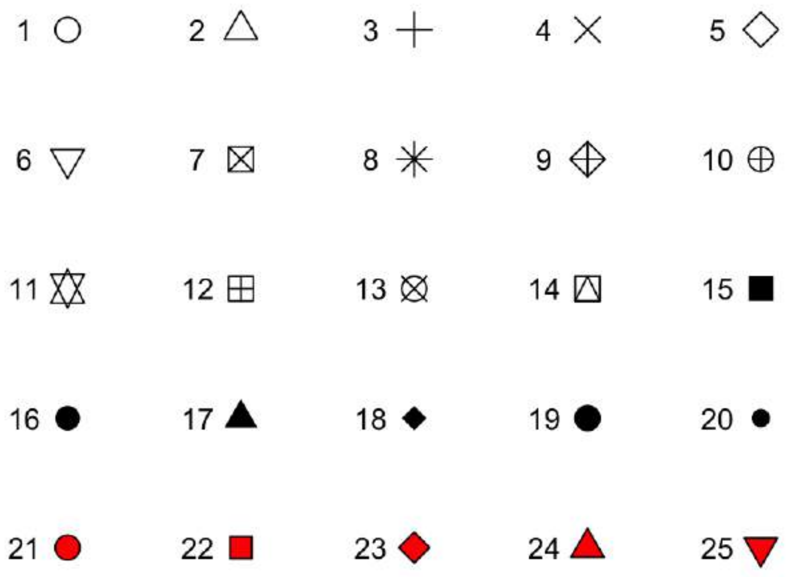


Figure 4.2: Examples of the shapes corresponding to different numeric choices for the `shape` aesthetic. For all examples, `color` is set to black and `fill` to red.

If you want to set color to be a constant value, you can do that in R using character strings for different colors. Figure 4.3 gives an example of a few of the different blues available in R. To find images that show all these named choices for colors in R, google “R colors” and search by “Images” (for example, there is a pdf here: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf).

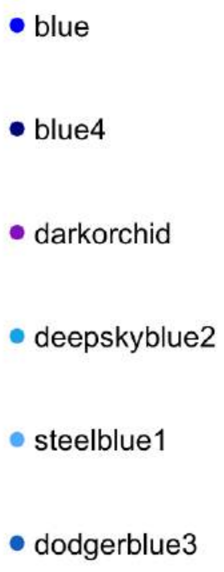


Figure 4.3: Example of a few available shades of blue in R.

Later we will cover additioal ways of handling colors in R, including different color palettes you can use. However, these “named” colors just shown can be a fast way to customize constant colors in R plots.

Other useful plot additions

There are also a number of elements besides geoms that you can add onto a `ggplot` object using `+`. A few that are used very frequently are:

Element	Description
`ggtitle`	Plot title
`xlab`, `ylab`	x- and y-axis labels
`xlim`, `ylim`	Limits of x- and y-axis

You can also use this syntax to customize plot scales and themes, which we will discuss later in this section.

Example plots

In this subsection, we’ll show a few more examples of basic plots created with `ggplot2`. For the example plots in this subsection, we’ll use a dataset in the `faraway` package called `nepali`. This gives data from a study of the health of a group of Nepalese children. You can load this data using:

```r
# install.packages("faraway") ## Uncomment if you do not have the faraway package
installed
library(faraway)
data(nepali)
```

Each observation in this dataframe represents a measurement for a child, including some physiological measurements like height and weight, and some children were measured multiple times and so have multiple observations in this data. Before plotting this data, we cleaned it a bit. We used tidyverse functions to select a subset of the columns: child id, sex, weight, height, and age. We also used the `distinct` function from `dplyr` to limit the dataset to the first measurement for each child.

```
nepali <- nepali %>%
  select(id, sex, wt, ht, age) %>%
  mutate(id = factor(id),
         sex = factor(sex, levels = c(1, 2),
                      labels = c("Male", "Female"))) %>%
  distinct(id, .keep_all = TRUE)
```

After this cleaning, the data looks like this:

```
head(nepali)
      id    sex   wt    ht age
1 120011   Male 12.8  91.2  41
2 120012 Female 14.9 103.9  57
3 120021 Female  7.7  70.1   8
4 120022 Female 12.1  86.4  35
5 120023   Male 14.2  99.4  49
6 120031   Male 13.9  96.4  46
```

We'll use this cleaned dataset to show how to use `ggplot2` to make histograms, scatterplots, and boxplots.

Histograms

Histograms show the distribution of a single variable. Therefore, `geom_histogram()` requires only one main aesthetic, `x`, which should be numeric. For example, to create a histogram of children's heights for the Nepali dataset (Figure 4.4), create a ggplot object with the data `nepali` and with the height column (`ht`) mapped to the ggplot object's x aesthetic. Then add a histogram geom:

```
ggplot(nepali, aes(x = ht)) +
  geom_histogram()
```

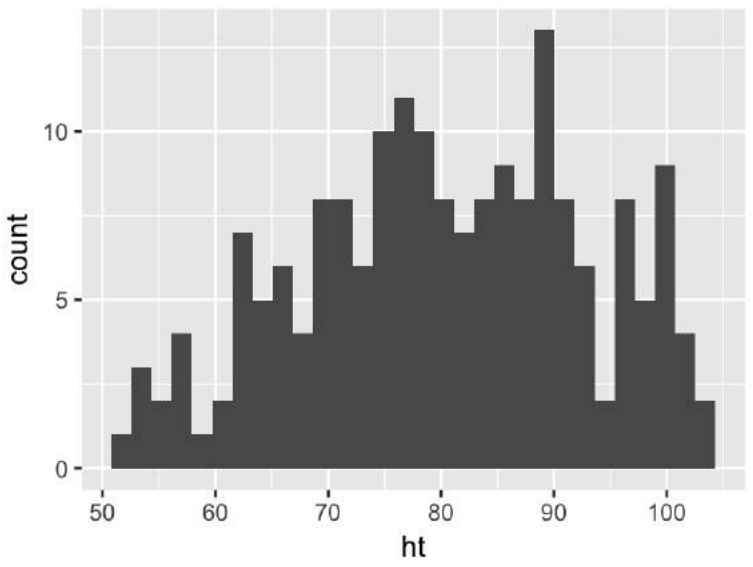


Figure 4.4: Basic example of plotting a histogram with `ggplot2`. This histogram shows the distribution of heights for the first recorded measurements of each child in the `nepali` dataset.

If you run the code with no arguments for `binwidth` or `bins` in `geom_histogram`, you will get a message saying "`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.". This message is just saying that a default number of bins was used to create the histogram. You can use arguments

to change the number of bins used, but often this default is fine. You may also get a message that observations with missing values were removed.
You can add some elements to this plot to customize it a bit. For example (Figure 4.5), you can add a figure title (`ggtitle`) and clearer labels for the x-axis (`xlab`). You can also change the range of values shown by the x-axis (`xlim`).

```
ggplot(nepali, aes(x = ht)) +
  geom_histogram(fill = "lightblue", color = "black") +
  ggtitle("Height of children") +
  xlab("Height (cm)") + xlim(c(0, 120))
```

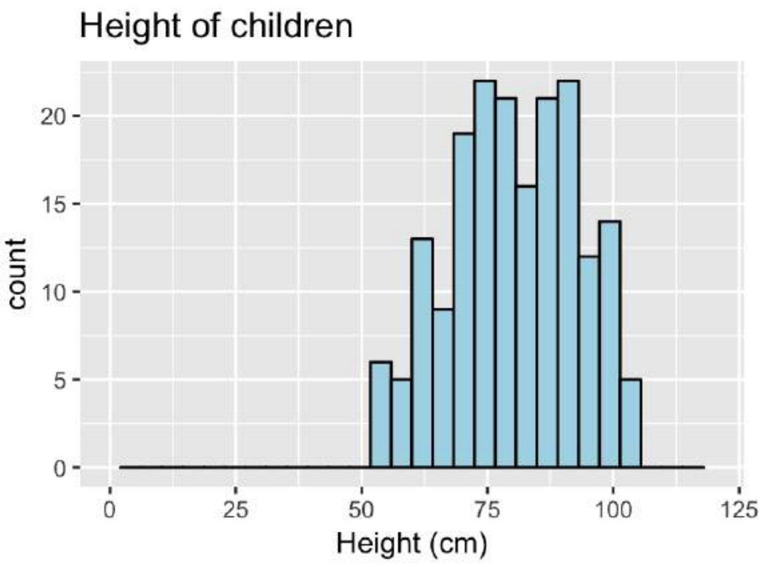


Figure 4.5: Example of adding ggplot elements to customize a histogram.

Note that these additional graphical elements are added on by adding function calls to `ggtitle`, `xlab`, and `xlim` to our ggplot object.

Scatterplots

A scatterplot shows the association between two variables. To create a scatterplot, add a point geom (`geom_point`) to a ggplot object. For example, to create a scatterplot of height versus age for the Nepali data (Figure 4.6), you can run the following code:

```
ggplot(nepali, aes(x = ht, y = wt)) +
  geom_point()
```

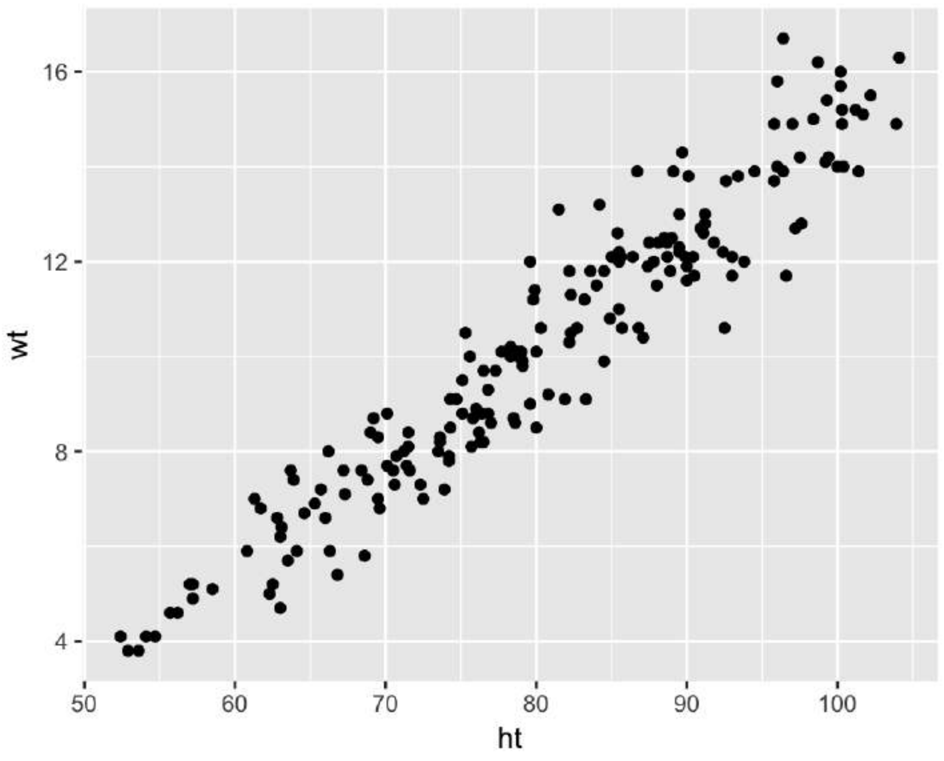


Figure 4.6: Example of creating a scatterplot. This scatterplot shows the relationship between children's heights and weights within the nepali dataset.

Again, you can use some of the options and additions to change the plot appearance. For example, to add a title, change the x- and y-axis labels, and change the color and size of the points on the scatterplot (Figure 4.7), you can run:

```
ggplot(nepali, aes(x = ht, y = wt)) +
  geom_point(color = "blue", size = 0.5) +
  ggtitle("Weight versus Height") +
  xlab("Height (cm)") + ylab("Weight (kg)")
```

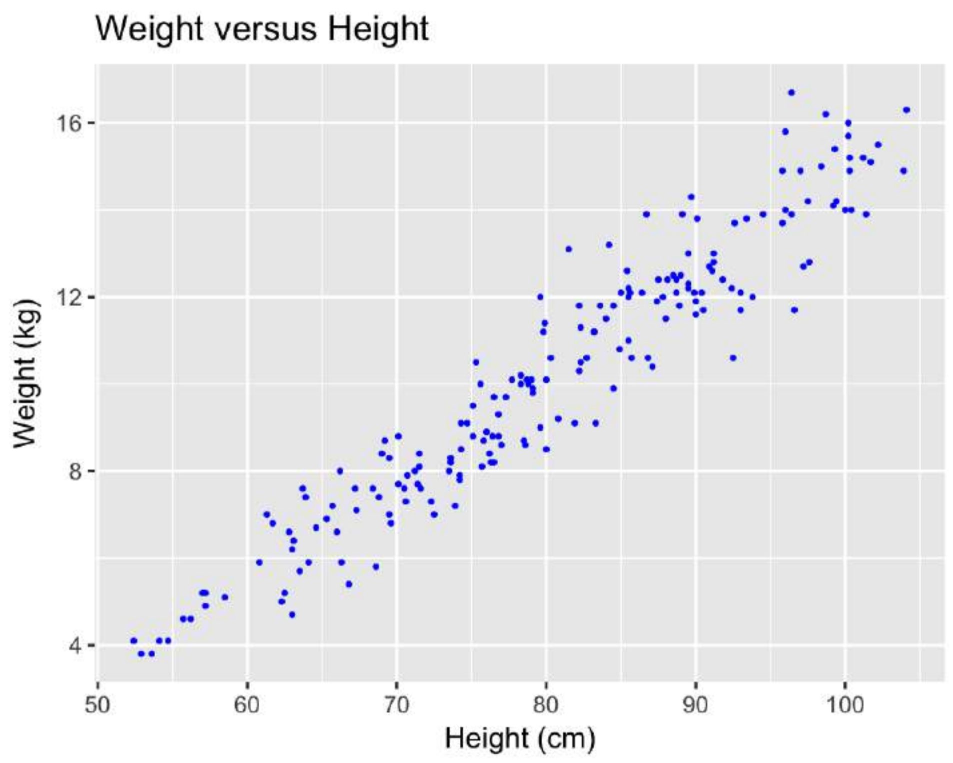


Figure 4.7: Example of adding ggplot elements to customize a scatterplot.

You can also try mapping a variable to the `color` aesthetic of the plot. For example, to use color to show the sex of each child in the scatterplot (Figure 4.8), you can run add an additional mapping of this optional aesthetic to the `sex` column of the `nepali` dataframe with the following code:

```
ggplot(nepali, aes(x = ht, y = wt, color = sex)) +
  geom_point(size = 0.5) +
  ggtitle("Weight versus Height") +
  xlab("Height (cm)") + ylab("Weight (kg)")
```

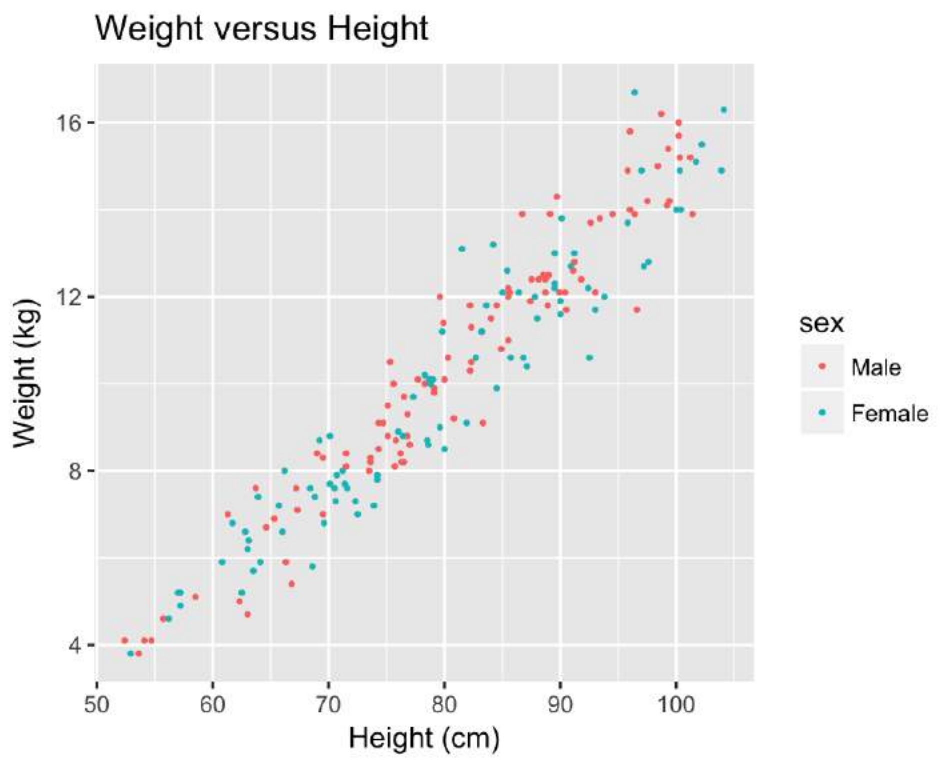


Figure 4.8: Example of mapping color to an element of the data in a scatterplot.

Boxplots

Boxplots are one way to show the distribution of a continuous variable. You can add a boxplot geom with the `geom_boxplot` function. To plot a boxplot for a single, continuous variable, you can map that variable to `y` in the `aes` call and map `x` to the constant `1`. For example, to create a boxplot of the heights of children in the Nepali dataset (Figure 4.9), you can run:

```
ggplot(nepali, aes(x = 1, y = ht)) +
  geom_boxplot() +
  xlab("")+ ylab("Height (cm)")
```

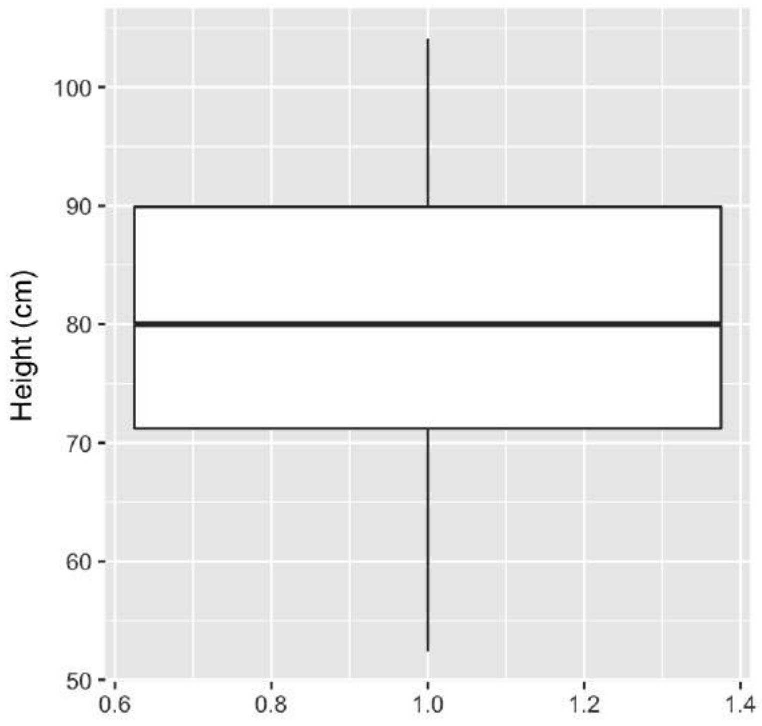


Figure 4.9: Example of creating a boxplot. The example shows the distribution of height data for children in the nepali dataset.

You can also create separate boxplots, one for each level of a factor (Figure 4.10). In this case, you'll need to map columns in the input dataframe to two aesthetics (x and y) when initializing the ggplot object The y variable is the variable for which the distribution will be shown, and the x variable should be a discrete (categorical or TRUE/FALSE) variable, which will be used to group the variable.

```
ggplot(nepali, aes(x = sex, y = ht)) +
  geom_boxplot() +
  xlab("Sex")+ ylab("Height (cm)")
```

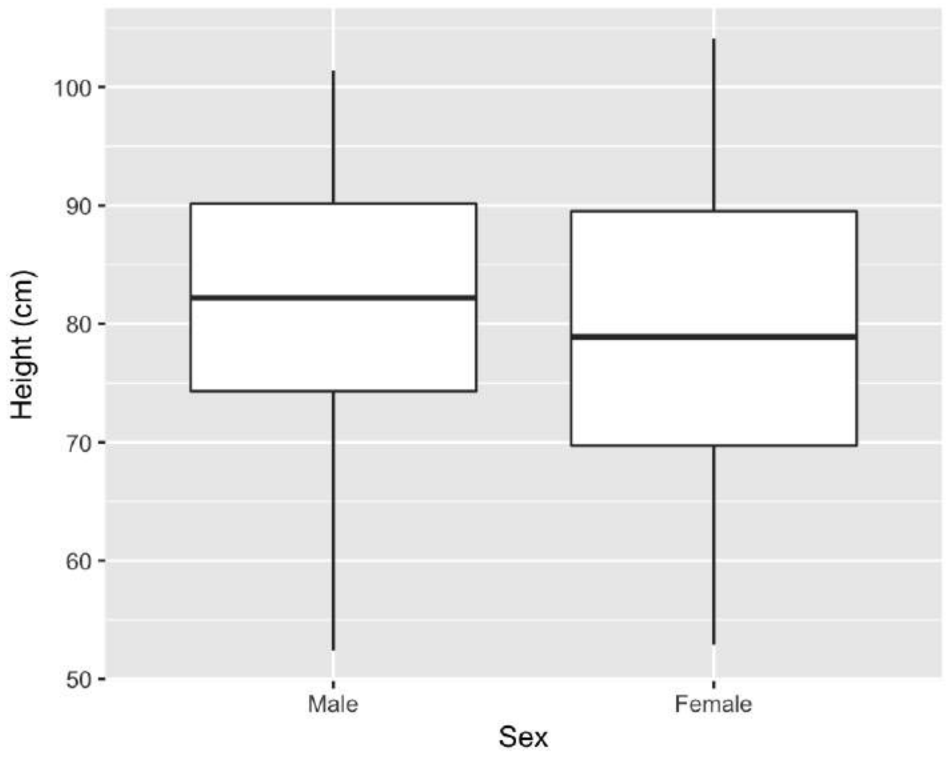


Figure 4.10: Example of creating separate boxplots, divided by a categorical grouping variable in the data.

Extensions of `ggplot2`

There are a number of packages that extend `ggplot2` and allow you to create a variety of interesting plots. For example, you can use the `ggpairs` function from the `GGally` package to plot all pairs of scatterplots for several variables (Figure 4.11).

```
library(GGally)
ggpairs(nepali %>% select(sex, wt, ht, age))
```

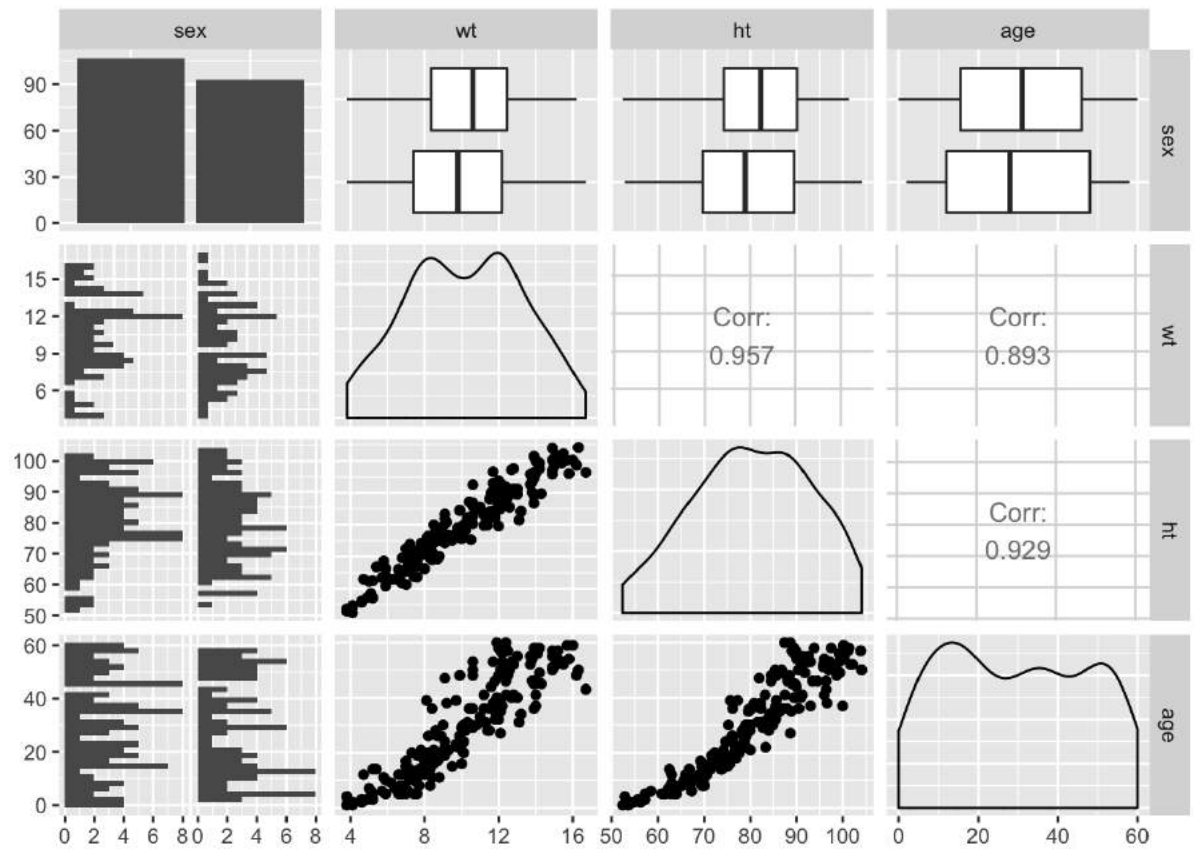


Figure 4.11: Example of using ggpairs from the GGally package for exploratory data analysis.

Notice how this output shows continuous and binary variables differently. For example, the center diagonal shows density plots for continuous variables, but a bar chart for the categorical variable.

See https://www.ggplot2-exts.org to find more `ggplot2` extensions. Later in this course, we will give an overview of how to make your own extensions.

Laboratory Procotol Developer and Supervisor(s) Information

Protocol Developer : Nanda R. Pradana Ratnasari
Email : nanda.ratnasari@i3l.ac.id

Supervisor(s)	email(s)
David Agustriawan	david.agustriawan@i3l.ac.id
Nanda R. Pradana Ratnasari	nanda.ratnasari@i3l.ac.id

Notice

1. Operate ONLY the computer assigned to you.
   aaa. If you have any troubleshoot please contact your supervisor or Building Management
   bbb. Do not rename files, adjust the dock size/icons, move items or files to the trash, or change the system preferences unless directed to do so
   ccc. Do not exchange keyboards, mice, or other equipment among the computers without notifying your supervisor
   ddd. Do not bring food or drinks into the lab unless it is in your backpack

2. Remain at your work center, respectful behavior promotes learning. Show integrity and respect for class materials. Responsibility is fundamental. Misused equipment will be replaced by those who damage it.

Session : 14

Date : December 10th, 2019

Laboratory :Bioinformatics laboratory

Overview

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

Objective

Students are expected to understand on how applying R development tools.

- In week 9, students are expected to understand how to run an Application of data manipulation as it is about fitting models to data.

Procedure

<u>Exercise</u>

1. For this part, you need to create a regression model using particular data provided in R.
    - Call the CO2 data
    - Discuss the summary and explain the scatterplot between "conc" and "uptake" variables
    - Determine the independent and dependent variables
    - Analyse the data using regression analysis for "conc" and "uptake" variables
    - Examine the results (including p_value for each intercept and the Adjusted R-square) and explain!

1. Generate random data with uniform distribution and try to create an algorithm to find the maximum value of the data.
- Generate 20 uniform random data with range from 0 to 10
- Create an algorithm to find the minimum value (using for and if function)
- Check your result using function *min()* to ensure that your code is running well

2. Generate random data with normal distribution and try to create an algorithm to find the maximum value of the data.
- Generate 10 normal random data with mean=0 and standar deviation=3
- Create an algorithm to find the maximum value (using for and if function)
- Check your result using function *max()* to ensure that your code is running well

4. For this section, you are asked to create a ggplot by following these steps
    - Call a data “diamons” provided in the package
    - Create a ggplot from the new data and differentiate by variable “color” using different colour
    - Cut the data to get rows 1-30
    - Create a ggplot from the new data and differentiate by variable “cut” using different shape

Application of Statistics and Data Sciences in Cancer

Nanda Rizqia Pradana Ratnasari, S.Si, MStat
Dr. rer.nat. Arli Aditya Parikesit
David Agustriawan.,PhD

1. Introduction

Cancer is one of diseases caused by the uncontrolled growth of abnormal cells that potentially spread or invade other cells and parts of the body. Cells of cancers develop when the normal cells of bodies lost their ability to quit thriving into new cells. When those cells grow uncontrollably, there might be changes in the DNA sequence, known as a mutation [1]. Cancer is also notable as the fastest evolving, an adapting and ever changing complex disease due to the genetic makeup [2] and it becomes one of four non-communicable diseases (NCD) as the causes of cancer are not completely understandable [3]. Numerous factors are noted to escalate the probability of occurrence of the disease, including the modifiable factors affected by behavior and the inherited conditions. As a multi-genic and multicellular disease, cancer can arise from all cell types with a multi-factorial etiology [8]. The increasing of risk factors may initiate or boost the growth of the disease and this uncontrolled growth can result in death [4]. In general, cancer is responsible for 14.6% of all human deaths and 63% of those numbers is reported to be death from developing countries [3]. The increasing number of the cases might be generated by several factors including population growth, ageing as well as the changing prevalence of certain causes of cancer linked to social and economic development [9].

Currently, there are more than 100 types of different cancers that attack human body and involve more than 500 genes in it [3]. The International Agency for Research on Cancer in 2012 reported that the number of death caused by the disease is estimated around 7.6 million worldwide due to increasing 12.7 million new cases per year [5, 6, 7]. This number increase to 9.6 million cancer deaths in 2018 with the disease burden rises to 18.1 million new cases [9]. Data gained from research and various resources show that there is an exponential growth of cancer. These data are complex and heterogeneous [3, 24]. Further report mentioned that one in 5 men and one in 6 women worldwide develop cancer during their lifetime, and one in 8 men and one in 11 women die from the disease. Worldwide, the total number of people who are alive within 5 years of a cancer diagnosis, called the 5-year prevalence, is estimated to be 43.8 million . Globally, nearly half of the new cases and more than 50% of the deaths caused by cancer occur in Asia [9].

Data Science which is an interdisciplinary subject involving statistics, scientific methods, processes and systems to extract knowledge from data has a good potential to revolutionize and improving health care, including advancing knowledge and information related to cancers. The medical areas has been generating and storing data for several decades in the form of clinical report, case study and hospital records, specifically in area of gene expression, next generation DNA sequence data, proteomics and metabolomics to discover effective drugs and treatments for the disease. Data Science, combined with other resource and knowledge, can help to resolve any challenges [2]. Statistics and Data Sciences become significant sectors to all areas of cancer. When data science can provide comprehensive information regarding characteristics and trend of population to present them in apprehensible explaination, statistics enable the data to be analyzed using valid methods, thus, the data and analysis can be an evidence for correct implementation [10]. Therefore, data science can elucidate broad and detail information about cancer and variables related to it, including explanation about medications, trend and patients. The information may be presented in visual or mathematical methods [11]. Moreover, statistics can account methods in cancer research, including the relationship of variables or covariates in cancer, how they react or affect each other, computational methods for examine particular treatments for cancer and predict future possibility of cancer and its medications, as well as explain how to create a model to do early diagnosis or forcasting about cancer cases [10, 11, 12]. Various medical project have been launched as part of implications and understanding of medical decision making. Data science enable researchers to analyze all kind of data to gain information and answer numerous medical question [24].

2. The Conceptual of This Study

This study will discuss about methods that might be applied in statistical analysis regarding cancer cases.

2.1 Data Science Analytic and Cancer

Data science in medical term encompasses many areas including Next Generation Sequencing (NGS) of the individual's genome, mRNA expression of normal and disease tissues, clinical trials, drug efficacy and medical records, for both in visuals or characters. The main goals of using data in cancer research are to transform the current approaches for diagnostics and treatments that have normally been done in wet-lab to a new dimension using data analytics by doing both identification and validations of the data. The dry analysis using data science enable to save time and budget of the research as data is ever available in many areas and can help people working effectively [2].

However, to apply big data technique and develop a sustainable healthcare system, researchers need to invest several amount of money and time [12].

Among numerous areas, cancer research area has huge amount of data science and occupied various statistical analysis [13-16]. Cancer data can be analyzed to find its distribution in each state or time. Variables that are affecting or affected by the disease can also be checked, in order to monitor the growth of the cells or the numbers of cases. In term of this study, regression analysis or analysis of variance can be conducted. For more advance study, the number of future case or the amount of cell in particular time is able to be predicted using time series or spatial analysis [2, 10, 12]. The high level of data science and statistical analysis is data collecting, modeling and analyzing to help in decision making. Those complexity and benefits of Data Science, however, it also faces some challenge with respect to its specific application such as: lacking of available variables, demanding reliable software, too much information that may not always been utilized [2, 23].

Source: David Wilks, Data at Government Digital Service, UK

Graph 1. Data Processing in Data Science and Statistical Analysis

2.2 Statistical Method in Cancer Research

The statistical Method that will be discussed in this study includes:

1. Kernel Density and Survival Analysis
2. Chi Square for Independence
3. Correlation
4. Significance Test (Hypothesis Test)
5. Regression
6. Analysis of Variance.

Table 1. Type of Statistical Analysis in Medical

Type of Analysis	The Purpose of The Test
Kernel Density and Survival Analysis	A non-parametric method to identify the probability distribution of survival analysis and estimate failure density
Chi Square for Independence	to investigate the relationship between categorical or qualitative variables

Correlation	to find and calculate the relationship and closeness between quantitative variables
Significance Test (Hypothesis Test)	to measure the strength of parameters, whether it is different to the given value or not
Regression	to find the relationship between independent and dependent variables

3. Kernel Density and Survival Analysis

Kernel density is also known as non-parametric method that can identify the probability distribution of survival analysis and estimate failure density [17]. Let be independent and identically distributed sample of random variables, then the nonparametric kernel density can be written as

where K is the kernel and is the estimate of the optimal bandwidth

The combination of the best Kernel values and optimal bandwidth needs to be tested to obtain the best kernel density estimation. Based on previous experiments, researchers gain formulas for Kernel and the bandwidth which are:

with

where SD is standard deviation and IQR is interquartile range.

The kernel density of survival function is given by

Meanwhile, mathematically the survival function can be defined as

denotes the survival time of event and describe the failure probability distribution.

The survival time of event defines how long a patient survives before they dead.

4. Chi Square Test of Independence

Chi square test of independence is also known as Pearson Chi square test [20]. *Chi square test of independence* is used to investigate the relationship between categorical or qualitative variables. In the cancer case, the analysis can be done to check the association between cancer and the location of the patients, gender or the group of age [18]. Statistical analysis of chi square provides information not only about significance of different between observations, but also can explain which variable account for the different. As one of the non-parametric test, chi square of independence test is noted as a distribution-free-test that does not require any assumption to be conducted [20]. In general a non-parametric test is utilized for data that have a level of measurements in nominal or ordinal, the sample size is not necessarily equal and violated one of parametric assumption such as the distribution of the data is skewed or kurtotic, it suffers from heterocesdasticity or data is no longer in the ratio or interval measurement. Meanwhile, the specific assumptions for the chi-square test are:

a. The data in the cell of observed table should be frequency
b. The level of variables should be mutually exclusive
c. Each subject may contribute data to one and only one cell in the chi square
d. The study group should be independent and the variables should be in nominal or ordinal level with multinomial probability distribution
e. The value in the expected cell should be 5 or more [18, 19].

To conduct this test we need to identify hypothesis whether there is association between the variables or not [18, 20].

The hypotheses for this test are:

: there is no association between two variables

: there is a significant association two variables

o conduct the test, the raw data are converted into contingency table, which is the table that consist of frequency for mutually exclusive of two categories. This table also known as observed table. The example of observed table can be found in table 1 of McHugh's paper [20] and Table 1 of Altham's paper [19].

Table 2. Result of the vaccination program

Health Outcome	Unvaccinated	Vaccinated	Total
Suffer from cancer	23	5	28
Suffer from other disease	8	10	18
No cancer	61	77	138
Total	92	92	184

Source: McHugh, M. (2013). The Chi-square test of independence. with modification

The table above show the frequency of patients observed to find the relationship between vaccination and the health outcome. With the data in the observed form, the researchers can proceed the values in the cells for calculating the . The formula for calculating the chi square is

where is the actual count of cases in each cell of the table and is the expected value obtained using this formula

represents the row marginal for that cell

represents the column marginal for that cell

represents the total sample size.

Further example and explanation regarding this value can be found in the Table 3 of (McHugh, 2013)[20].

Table 3. Table of Expected Value and Chi-Square cell for the vaccination program

Health Outcome	Unvaccinated	Vaccinated
Suffer from cancer	13.92 (5.92)	12.57 (4.56)
Suffer from other disease	8.95 (0.10)	9.05 (9.05)
No cancer	69.12 (0.95)	69.88 (0.73)

Source: McHugh, M. (2013). The Chi-square test of independence. with modification

Once the cell $\chi 2$ values have been calculated, they are summed to obtain the $\chi 2$ statistic for the table. To conclude the decision, the statistics obtained from calculation should be compared to table with degree of freedom (number of rows-1)*(number of columns -1). The null hypothesis is rejected if the statistics is bigger than table or p-value of the is less than the level of significance.

Rejecting the null hypothesis means that there is no association or relation between variables [18, 19].

Based on values in the cells of table 2, the total value of is 12.35 which is less than . Therefore, there is enough evidence to reject . Thus, there is association between health outcomes with vaccination program.

5. Correlation

Correlation becomes another analysis to find and calculate the relationship and closeness between quantitative variables. The linear correlation coefficient measures the degree association between variables. The negative sign in the measurements represents negative relation between observed feature, one variable may increase while the other variable decrease [27]. Two correlation coefficient normally used in application research are noted as Pearson's Correlation and Spearman's Rank Correlation [28].

In general Pearson's Correlation Coefficient formula is denoted below:
where are measures of variable and is the number of observation. The formula or also can be written as

The correlation values has range from -1 to +1 and both values represent very strong association between two variables. Meanwhile, the 0 (zero) value or any number close to zero that be obtained from calculation indicates the no-association between variables [27]. Gogtay and Thatte in [29] explained the interpretation of correlation coefficient based on the value.

Source: Gogtay and Thatte (2017, p. 79) and Senthilnathan (2019)

Graph 2. The interpretation of correlation coefficient

Source: Gogtay and Thatte (2017, p. 79) and Senthilnathan (2019)

Graph 2. The interpretation of correlation coefficient

Correlation is not necessarily causation effects. However, when it is known that there is causation between independent and dependent variables, correlation can be analyzed using linear correlation coefficient or coefficient of determination [29]. This value provides information on how much accuracy and variation of dependent variable respect to independent variables.

Correlation analysis in cancer case has been applied by Lin and Huang [30] to find the connection between molecular subtype which have similar phenotype with their transcriptional profile. The data analyzed in the study conducted by Lin and Huang are the gene expression. The other research using correlation analysis in cancer case was conducted by Ling at all to find the association between Gene Expression and Cancer Diagnosis [31].

6. Significance Test (Hypothesis Test)

A significant test enables researchers to measure the strength of parameters of events or experiments. It can analyze whether the difference in the observation is small enough to occur by chance if there is no different in population. To carry out the test of significance or hypothesis test, a researcher should suppose that there is no different between two treatments [32]. The method of hypothesis testing is to determine the likelihood if the statement is true. The test is normally related to mean or variance [34].

The hypothesis that there is no different is called the null hypothesis, while the alternative hypothesis states that there is different between the treatments. The direction of the alternative hypothesis explains type of the test. Two-side test will be conducted if probability of extreme values in both tails of normal curve is used [32]. In two-sided test, the null hypothesis states that the parameters equals to a provided value. Meanwhile, in one-sided test, the null hypothesis will present that the parameter is less or bigger than the particular value provided in the test [34].

The general procedure for a significant test is as follow:

a. Determining the null and alternative hypothesis
b. Calculating the value of the test statistics
c. Referring to the test statistics to determine which distribution should be followed; then comparing the statistics value with the value coming from the distribution
d. Finding the probability of the test statistics
e. Concluding that the data se consistent with null hypothesis or not: either reject or fail to reject the null hypothesis[32, 34]

In term of hypothesis concerning mean, there are two type of statistical test, which are z-test and t-test. Z-test is used for the mean of population, when we have big sample (greater than 30), when the data is normally distributed and the standard deviation is known. Meanwhile, the t-test should be conducted when the standard deviation is unknown or when the number of data is small [35]. For single mean test, statistic value can be derived from this formula:
Meanwhile, the equality of mean test can be conduct using statistic value:

Test decision can be determined by comparing the statistic value with the critical value. The null hypothesis is rejected if for one-sided test or for two-sided test. Similar to the t-test, where the null hypothesis is rejected if for one-sided test and for two-sided test, where df is degree of freedom which is obtained from [32-35].

If the standard deviation is unknown, it will be estimated using sample standard deviation with the formula [33]:

Graph 3. Normal curve for conducting significant test decision

The hypothesis test can be conducted in are of cancer, for example to check the efficacy or effectiveness of drugs or treatments for curing a particular disease. Another research might be undertaken to investigate the effect of characteristic of patients to their survival power. Some applications of hypothesis test in cancer areas can be found in [1, 33].

In statistics test of hypothesis, a researcher needs to consider two kinds of error in order to make a good decision for their research. The type I error () declare to reject when it is actually true, while type II error () state to accept when it is really false. Hence, the power is the chance of not making a type II error and noted as () [32, 33].

Table 4. Table of Error Type in Statistical Test

Hypothesis	true	false
true	Type I error	
false		Type II error

Graph 4. The area of Type I Error in Normal Distribution Curve

Graph 5. The area of Type II Error in Normal Distribution Curve

Graph 6. The area of power

7. Regression

Regression analysis is one of the most frequent statistical methodology conducted in various areas to find the relationship between independent and dependent variables [36]. It is also known as a method for estimating parameters and numerical association between variables. In regression, people are interested to predict one variable respecting to other variables. The variables that is explaining the other variables are known as explanatory or predictor (independent) variable. Meanwhile, the variables explained by the predictor are known as outcome or dependent variables [32]. Coefficient in a regression model can be estimated using a method called Ordinary Least Square (OLS). The concept behind this method is to

minimize the square error between the estimated value and the real parameter of the data [37]. In general, a regression model can be written as
where :
is dependent variable
are independent variable
are parameters

The parameter estimation can be conducted by considering the data as metrics. Therefore, the function for the least square method can be noted as
where

The next step is differentiating with respect to in which . This has to be done to ensure that the value of is minimum.
Hence, the estimator will be
A good regression analysis and model should satisfy the classic assumptions that are

a. The constant variance of residuals. When this assumption is satisfied, the error will indeed be random and independently distributed.
b. The residual is normally distributed
c. Multicollinearity does not exist [32 , 37]

The assumption checking can be conducted using residual plot (scatterplot of the error) of residual versus fitted model. The residuals are independent and have constant variance if the dots in the scatter plot are randomly spreading along the axis. Furthermore, the normality assumption is satisfied if the dots follow the linear line in the QQ-plot [36, 37].

Graph 7. Scatterplot of regression residuals for assumption checking

The multicollinearity is the condition where there is strong correlation between independent variables thus causing duplication of similar information in the estimation. Duplication of information in the estimator causes inaccurate model prediction and forcasting [27]. VIF is the measure of detecting level of multicollinearity between the predictors.

Where explains the correlation coefficient between the predictors

Table 5. VIF Criteria

Criteria	Intepretation of Multicollinearity

VIF=1 (implies no correlation)	null
1<VIF<5 (correlated)	lower level
5VIF< (highly correated)	high level
VIF (perfectly correlated)	perfect muticollinearity

Research conducted using regression analysis in cancer area can be found in research by *Shafi et all* [37].

References

[1] Muthuvel, M., Keerthika, V. and Poobalan, C. P. S. C. (2018). Statistical Analysis in Cancer Data. https://www.researchgate.net/publication/326710902

[2] Deekshita, S. (2017). Data Science and Cancer: An Approach to the Challenges Involved International Journal of Computer Science & Engineering Technology (IJCSET) 8(07): 303-307

[3] Pavlopoulou, A., Spandidos, D. A. and Michalopoulos, I. (2015). Human Cancer Database (Review). Oncology Report 33: 3-18, 2015

[4] American Cancer Society. (2019). Cancer Facts and Figures 2019. https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2019/cancer-facts-and-figures-2019.pdf

[5] International Agency for Research on Cancer (IARC): GLOBOCAN 2008, Cancer incidence and mortality worldwide. Lyon, France: IARC. 2010.

[6] Jemal A, Bray F, Center MM, Ferlay J, Ward E, Forman D: Global cancer statistics. CA Cancer J Clin. 2011; 61: 69-90.

[7] Ferlay J, Shin HR, Bray F, Forman D, Mathers C, Parkin DM: Estimates of worldwide burden of cancer in 2008: GLOBOCAN 2008. Int J Cancer 2010; 127: 2893-2917.

[8] Baskar, R., Lee, K. A., Yeo. R. and Yeoh, K. W. (2012). Cancer and Radiation Therapy: Current Advances and Future Directions. *International Journal in Medical Sciences* 9(3):193-199. doi: 10.7150/ijms.3635

[9] International Agency for Research on Cancer, World Health Organization (IARC). (2018). Latest global cancer data: Cancer burden rises to 18.1 million new cases and 9.6 million cancer deaths in 2018. Lyon, France: IARC.

[10] Breslow, N. E. and Day, N. E. (1980). Statistical Methods in Cancer Research. *IARC Scientific Publication 1(32)*

[11] Zhu, Y. and Xiong, Y. (2015). Defining Data Science- Beyond the study of the rules of the natural world as reflected by data. https://arxiv.org/ftp/arxiv/papers/1501/1501.05039.pdf

[12] Narayanan, R. and Makler, A. (2016). Big Data Analytic and Cancer. *MOJ Proteomics and Bioinformatics* 4(2): 00115. DOI: 10.15406/mojpb.2016.04.00115

[13] Coates J, Souhami L, El Naqa I (2016) Big Data Analytics for Prostate Radiotherapy. Front Oncol 6: 149.

[14] Swift SL, Stojdl DF (2016) Big Data Offers Novel Insights for Oncolytic Virus Immunotherapy. Viruses 8(2): E45

[15] Yang Y, Dong X, Xie B, Ding N, Chen J, et al. (2015) Databases and web tools for cancer genomics study. Genomics Proteomics Bioinformatics 13(1): 46-50

[16] Kim ES (2015) The Future of Molecular Medicine: Biomarkers, BATTLEs and Big Data. Am Soc Clin Oncol Educ Book 22-27

[17] Kottabi, Z. (2012). Statistical Modeling and Analysis of Breast Cancer and Pancreatic Cancer. *Graduate Thesis and Dissertation*. http://scholarcommons.usf.edu/etd/4350

[18] Adamu, P. I., Oguntunde, P. E, and Okagbue, H. I. (2018). Statistical Data Analysis of Cancer Incidences in Insurgency Affected States in Nigeria. *Elsevier* Data in Brief 18 p. 2029-2046

[19] Altham, P. (2005). Contingency Table. 10.1002/0470011815.b2a10016.

[20] McHugh, M. (2013). The Chi-square test of independence. Biochemia medica. 23. 143-9. 10.11613/BM.2013.018.

[21] Magalhaes, L. G., Ferreira, L. L. G. and Andricopulo, A. D.(2018). Recent Advances and Perspectives in Cancer Drug Design. *Anais da Academia Brasileira de Ciências* 90(1 Suppl. 2): 1233-1250

[22] Kaplan, W. (2004). Priority Medicines for Europe and the World "A Public Health Approach to Innovation". https://www.researchgate.net/publication/249995116_Priority_Medicines_for_Europe_and_the_World_A_Public_Health_Approach_to_Innovation_Update_on_the_2004_Background_Paper

[23] Das, S. R. (2016). Data Science: Theories, Models, Algorithms, and Analytics. https://srdas.github.io/Papers/DSA_Book.pdf

[24] Ibnouhsein I, Jankowski S, Neuberger K, Mathelin C. The Big Data Revolution for Breast Cancer Patients. Eur J Breast Health 2018; 14: 61-62

[25] Reuben, S. H., Milliken, E. L. and Paradis, L. J. (2012). The Future of Cancer Research: Accelerating Scientific Innovation. President's Cancer Panel Annual Report 2010-2011. U.S. DEPARTMENT OF HEALTH AND HUMAN SERVICES, National Institutes of Health National Cancer Institute

[26] Lee, K. A., Yeoh, K.W., and Yeo, R. (2012). Cancer and Radiation Therapy: Current Advances and Future Directions. *International Journal of Medical Science* 9(3):193-199. doi: 10.7150/ijms.3635

[27] Senthilnathan, S. (2019). Usefulness of Correlation Analysis. https://www.researchgate.net/publication/334308527

[28] Hauke, J. and Kossowski, T. (2011), Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data, ***Questiones Geographicae***, 30(2), pp. 87-93 (doi: http://dx.doi.org/10.2478/v10117-011-0021-1).

[29] Gogtay, N. J. and Thatte, U. M. (2017), Principles of correlation Analysis, ***Journal of The Association of Physicians of India***, 65 (March), pp. 78-81

[30] Lin P, Huang Z (2013) Correlation Analysis Connects Cancer Subtypes. PLoS ONE 8(7): e69747. doi:10.1371/journal.pone.0069747

[31] Ling, B., Chen, L., Liu, Q. and Yang, J. (2014). Gene Expression Correlation for Cancer Diagnosis: A Pilot Study. BioMed Research International Volume 2014, Article ID 253804, 6 pages http://dx.doi.org/10.1155/2014/253804.

[32] Bland, M. (1995). An Introduction t Medical Statistics. *Oxford Medical Publication*.

[33] Dunn, O. J. and Clark, V. A. (2009). Basic statistics: a primer for the biomedical sciences. http://www.academia.dk/BiologiskAntropologi/Epidemiologi/PDF/Basic_Statistics__A_Primer_for_the_Biomedical_Sciences_4th_Ed.pdf

[34] Massey, A. and Miller, S. J. Test of Hypotheses Using Statistics. https://web.williams.edu/Mathematics/sjmiller/public_html/BrownClasses/162/Handouts/StatsTests04.pdf

[35] Bluman, A. (2009). ELEMENTARY STATISTICS: A STEP BY STEP APPROACH, SEVENTH EDITION. New York: USA

[36] Mooi, E. (2016). Regression Analysis. DOI: 10.1007/978-3-642-53965-7_7. https://www.researchgate.net/publication/300403700

[37] Shafi, M. A. ad Rusiman, M.S. (2016). The use of Linear Statistical Model to Predict Tumour Size of Colorectal Cancer. Journal of Science and Technology, Vol. 8 No. 2 (2016) p. 1-5