Intro about R, github and stuff

Vita Tomkutė

2019, November

Class Structure and Organization

- Ask questions at any time. Really!
- Collaboration is encouraged

We talk, some demo, some exercises

Why R?

And not matlab, python, SSPS, ...

Why R?

And not matlab, python, SSPS, ...

- Open source and free
- Flexible – can create whatever you need (programming language)
- Works on different OS
- Very convenient visualizations
- Wide R user community
- …

Why R?

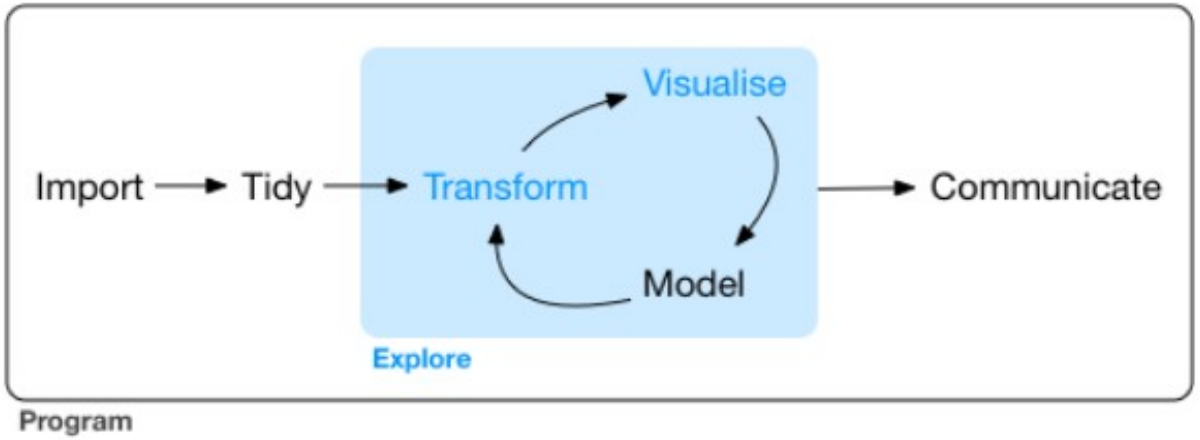
And not matlab, python, SSPS, ...

- Open source and free
- Flexible – can create whatever you need (programming language)
- Works on different OS
- Very convenient visualizations
- Wide R user community
- …

R is a tool !

Today’s goals:

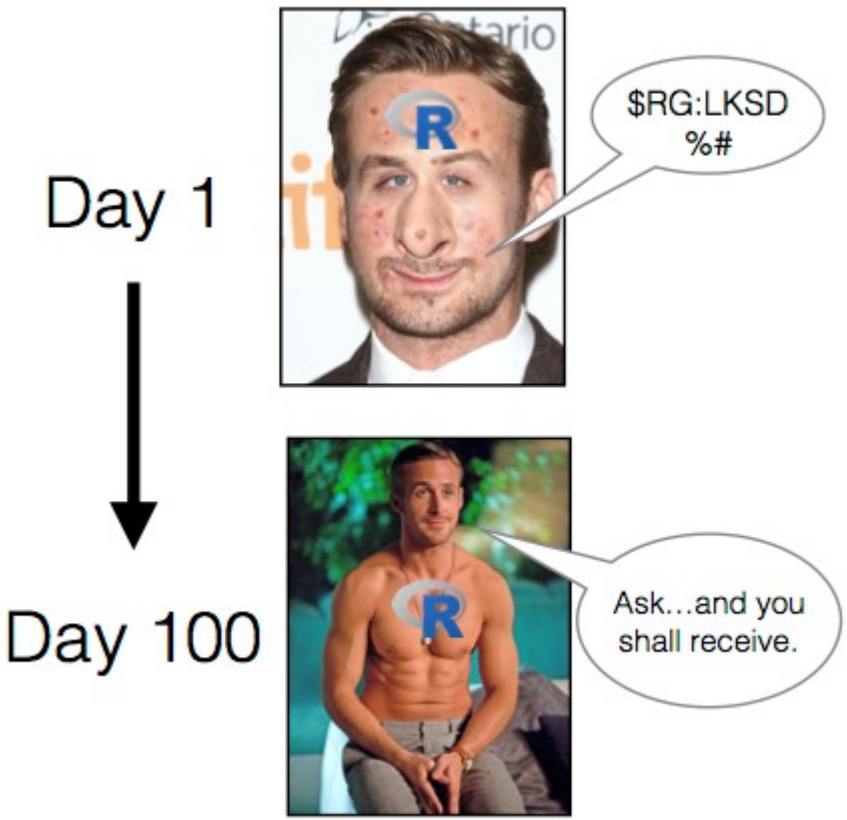
- 1) Familiarize with the tool
- 2) Exploratory data analysis



R...

https://bookdown.org/ndphillips/YaRrr/images/rrelationship.png

After some time... it gets better

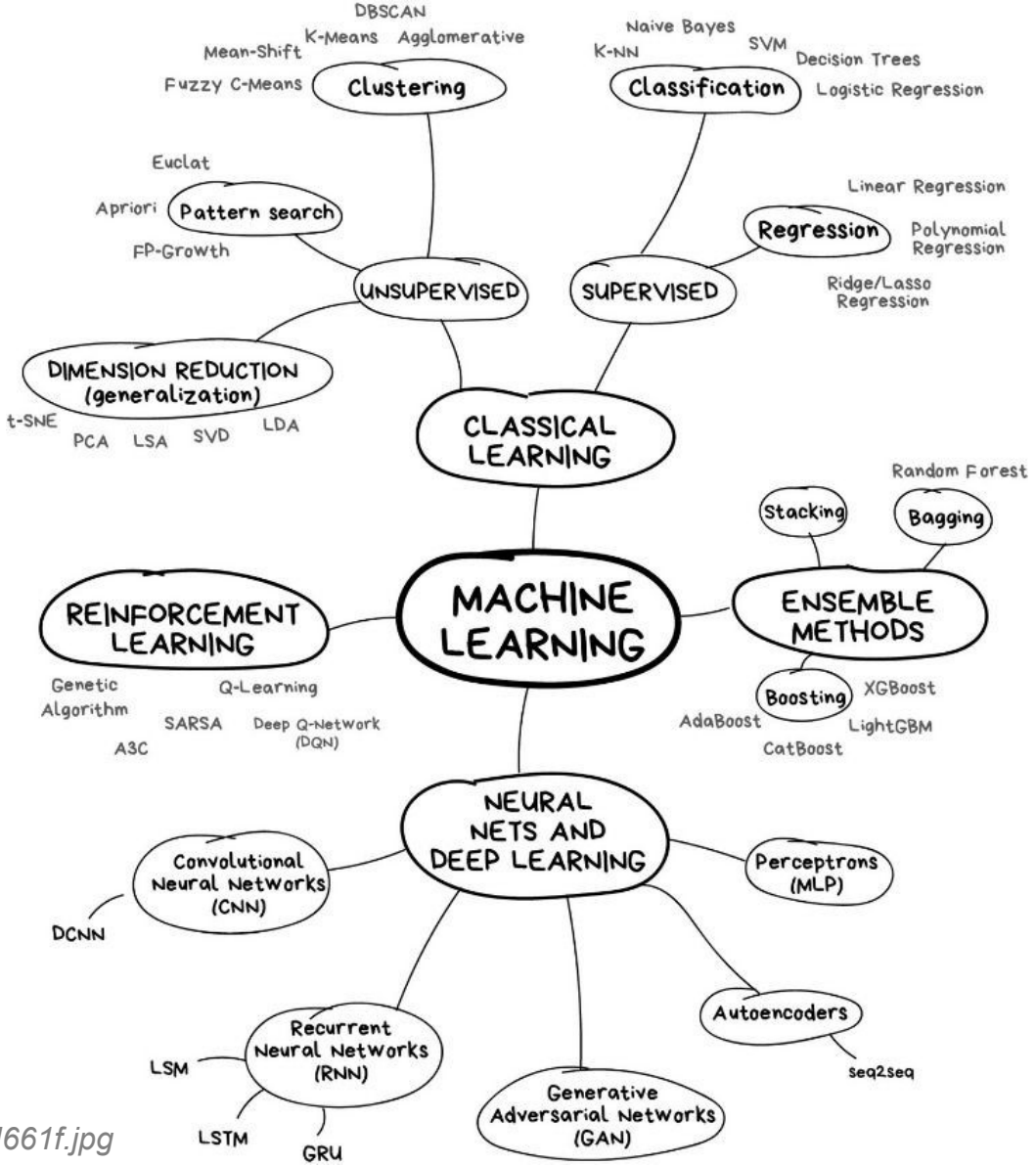


https://bookdown.org/ndphillips/YaRrr/images/gosling.png

Methods

There is no BEST method.

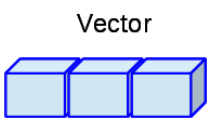
- Depends on the:
  - Question/Goal
    (explore, understand, predict)
  - Data type
    (labeled or unlabeled data, continuous, discrete, categorical data)

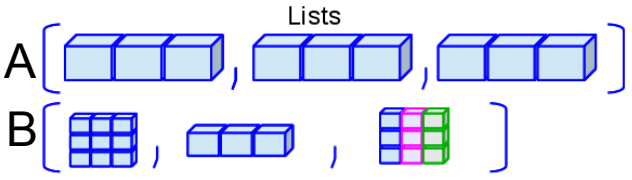


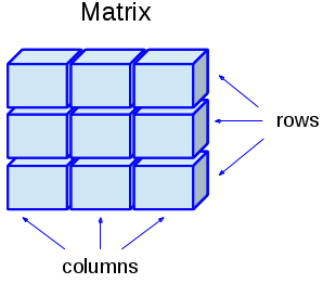
https://i.pinimg.com/originals/08/1c/49/081c49abac0550512300bf78bcbd661f.jpg

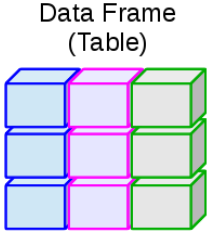
Data types in R

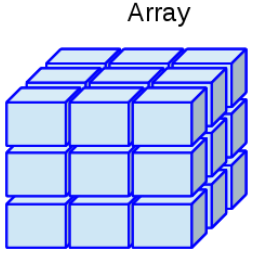
	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	







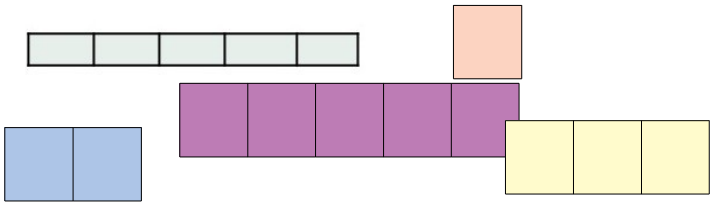




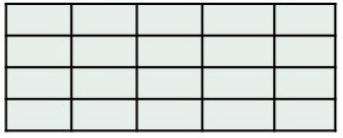
Heterogenous vs Homogenous

Variables	Example
integer	100
numeric	0.05
character	“hello”
logical	TRUE
factor	“Green”

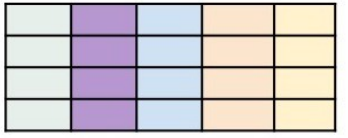
Vector



Matrix



Data frame



tidyverse

Components

The tidyverse is a collection of R packages that share common philosophies and are designed to work together.

https://hbctraining.github.io/Intro-to-R/lessons/tidyverse_data_wrangling.html

ggplot2

What Is The Grammar Of Graphics?

The basic idea: independently specify plot building blocks and combine them to create just about any kind of graphical display you want.

Building blocks of a graph include:

1) data
2) aesthetic mapping
3) geometric object
4) faceting
5) statistical transformations
6) coordinate system
7) theme

- 1st layer


```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
...
50           5.0         3.3          1.4         0.2     setosa
51           7.0         3.2          4.7         1.4 versicolor
52           6.4         3.2          4.5         1.5 versicolor
53           6.9         3.1          4.9         1.5 versicolor
...
100          5.7         2.8          4.1         1.3 versicolor
101          6.3         3.3          6.0         2.5  virginica
102          5.8         2.7          5.1         1.9  virginica
103          7.1         3.0          5.9         2.1  virginica
...
150          5.9         3.0          5.1         1.8  virginica
```

Species Sepal.Length Sepal.Width Petal.Length Petal.Width

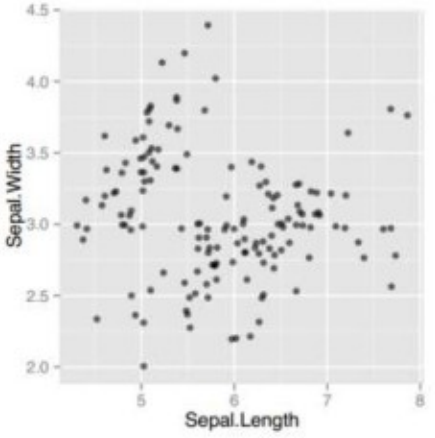
X Y

- 2nd layer


```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
...
50           5.0         3.3          1.4         0.2     setosa
51           7.0         3.2          4.7         1.4 versicolor
52           6.4         3.2          4.5         1.5 versicolor
53           6.9         3.1          4.9         1.5 versicolor
...
100          5.7         2.8          4.1         1.3 versicolor
101          6.3         3.3          6.0         2.5  virginica
102          5.8         2.7          5.1         1.9  virginica
103          7.1         3.0          5.9         2.1  virginica
...
150          5.9         3.0          5.1         1.8  virginica
```

```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_jitter(alpha = 0.6)
```

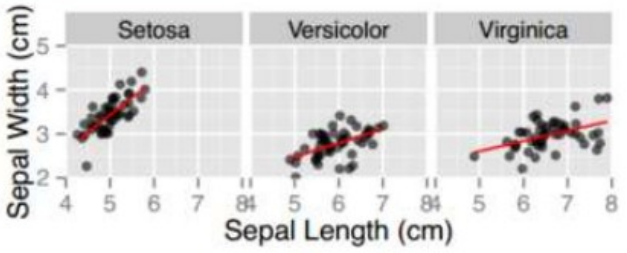
- 3rd layer



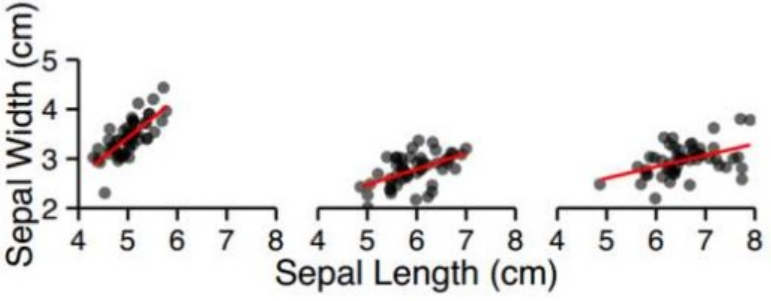
These 3 layers are necessary for all ggplot2 plots

- 6th layer


```
> levels(iris$Species) <- c("Setosa", "Versicolor", "Virginica")
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_jitter(alpha = 0.6) +
    facet_grid(. ~ Species) +
    stat_smooth(method = "lm", se = F, col = "red") +
    scale_y_continuous("Sepal Width (cm)",
                       limits = c(2,5),
                       expand = c(0,0)) +
    scale_x_continuous("Sepal Length (cm)",
                       limits = c(4,8),
                       expand = c(0,0)) +
    coord_equal()
```



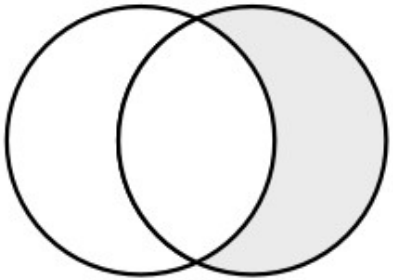
- 7th layer



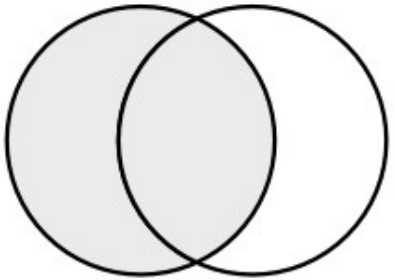
package(dplyr)

- select() extracts columns and returns a tibble.
- arrange() changes the ordering of the rows.
- filter() picks cases based on their values.
- mutate() adds new variables that are functions of existing variables.
- rename() easily changes the name of a column(s)
- summarise() reduces multiple values down to a single summary.
- group_by()

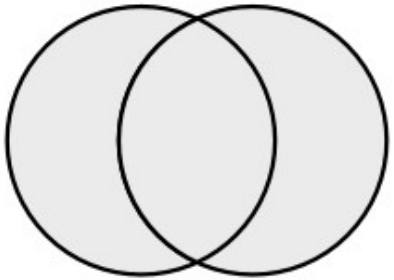
Logical operators



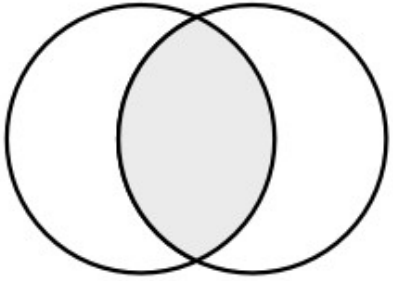
y & !x



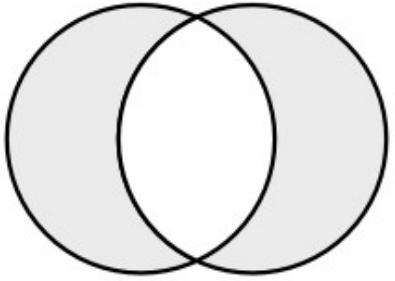
x



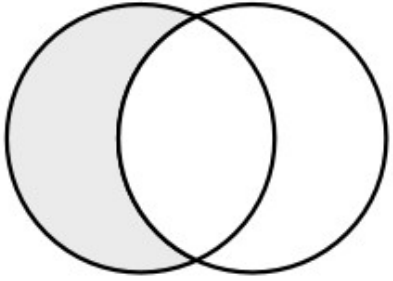
x | y



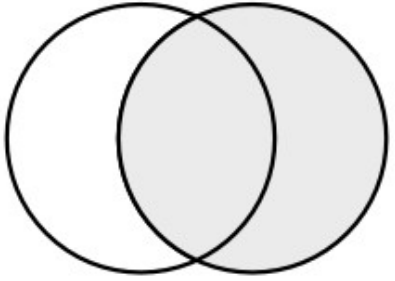
x & y



xor(x, y)

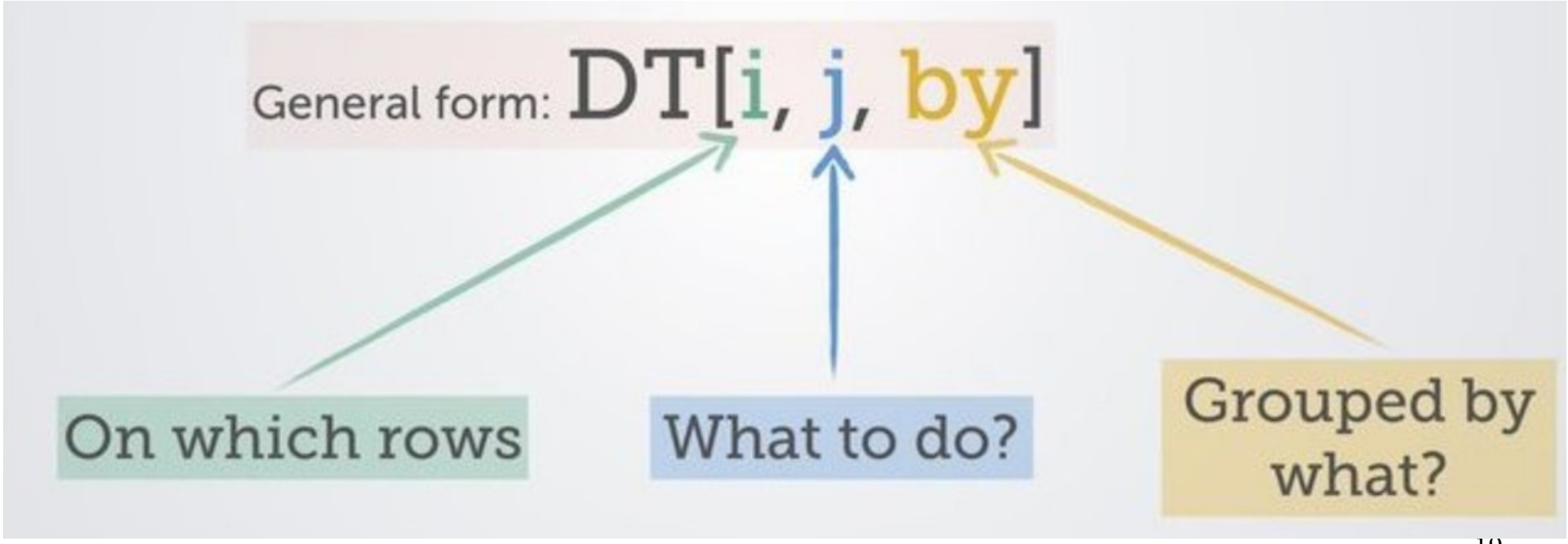


x & !y

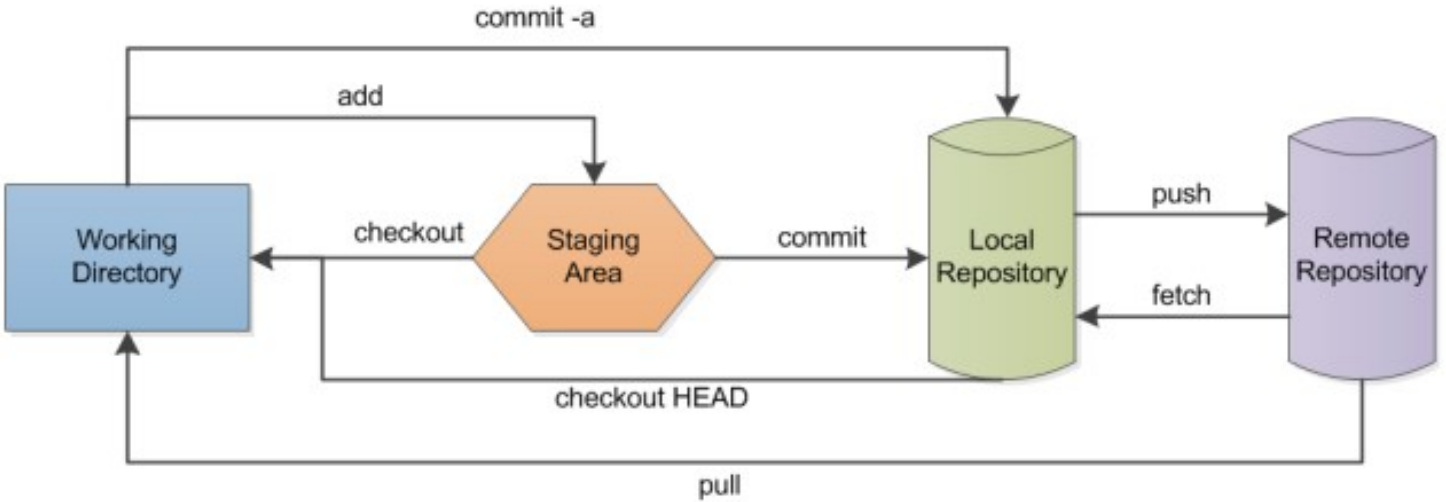


y

package(data.table)



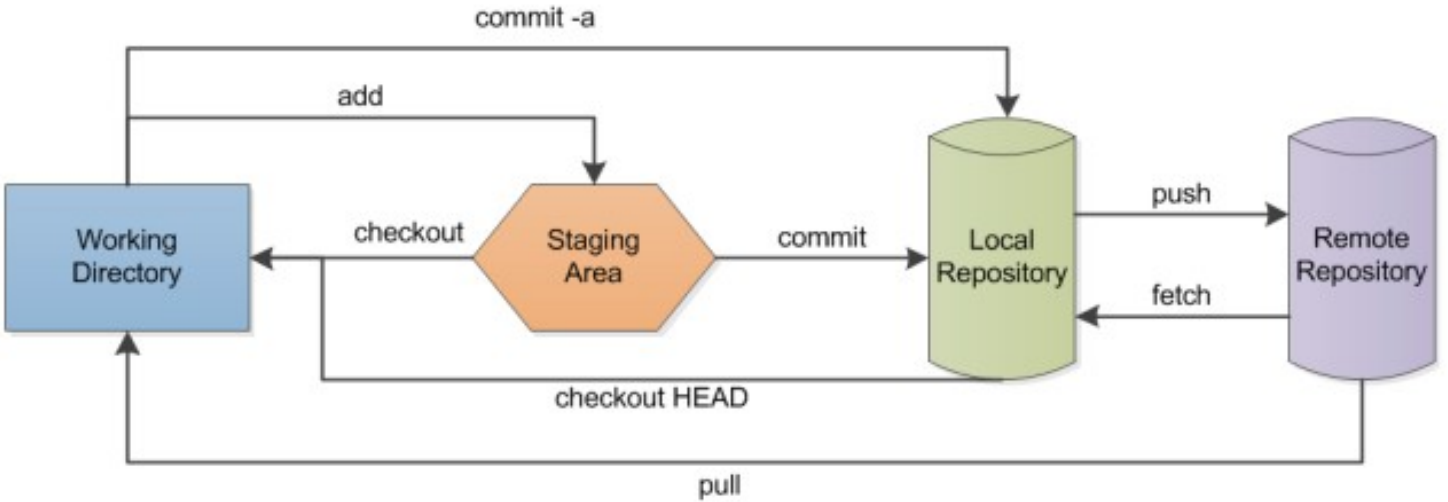
Git and Github



git

`clone`: copy remote repository to your working directory
`status`: check which files are staged, changed, ect.
`add`: add a file to the staging area
`commit`: commit your files from staging to local repository i.e. like a save for your the changes you made. Commits are recorded and you can go back to committed stages afterwards
`checkout HEAD`: return to the last commit
`checkout`: unstage the files (get them back from the staging area)
`push`: send files to the remote repository
`fetch`: get the latest changes from an online repository without merging them in
`pull`: get the latest changes from online repository and merge them to your working directory (so you actually get the files)
`init`: initiate local repository without connection to remote directory
`branch`: parallel version of local repository. It is contained within local repository but is different from primary or `master` branch
`log` or `hist` and other….

Git and Github



.gitignore

```
#ignore those files
*.RDS
*.bam
*.sam
*.fasta
```

git

`clone`: copy remote repository to your working directory
`status`: check which files are staged, changed, ect.
`add`: add a file to the staging area
`commit`: commit your files from staging to local repository i.e. like a save for your the changes you made. Commits are recorded and you can go back to committed stages afterwards
`checkout HEAD`: return to the last commit
`checkout`: unstage the files (get them back from the staging area)
`push`: send files to the remote repository
`fetch`: get the latest changes from an online repository without merging them in
`pull`: get the latest changes from online repository and merge them to your working directory (so you actually get the files)
`init`: initiate local repository without connection to remote directory
`branch`: parallel version of local repository. It is contained within local repository but is different from primary or `master` branch
`log` or `hist` and other….