R programming

Philip J Cwynar

University of Pittsburgh
School of Information Sciences
and Intelligent Systems Program

Background

R is a programming language and software environment for statistical analysis, graphics representation and reporting.

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems like Linux, Windows and Mac.

Intro

The following important features of R:

R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities.

R has an effective data handling and storage facility,

R provides a suite of operators for calculations on arrays, lists, vectors and matrices.

R provides a large, coherent and integrated collection of tools for data analysis.

R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

R is world's most widely used statistics programming language.

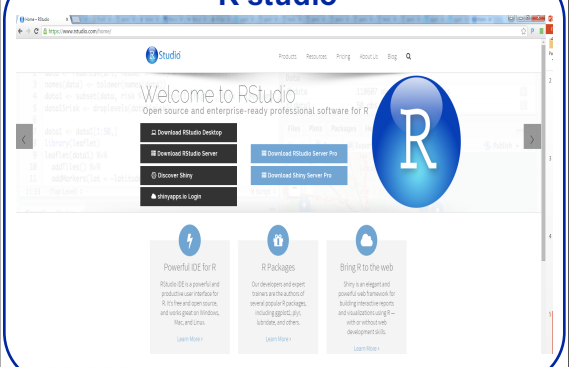
It's the # 1 choice of data scientists and supported by a vibrant and talented community of contributors. R is taught in universities and deployed in mission critical business applications.

Download and install R here:

https://www.r-project.org/

https://cran.cnr.berkeley.edu/

R studio

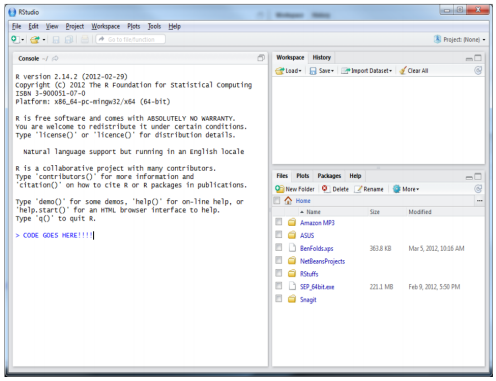


R studio

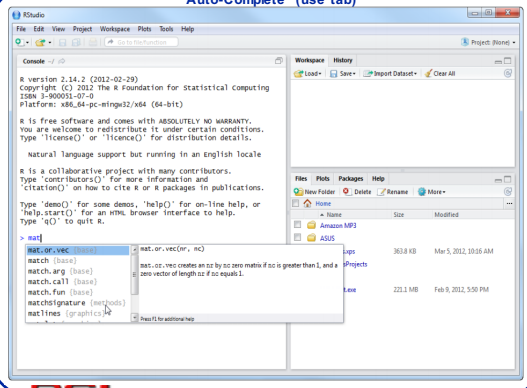
https://www.rstudio.com/

https://www.rstudio.com/products/rstudio/download/

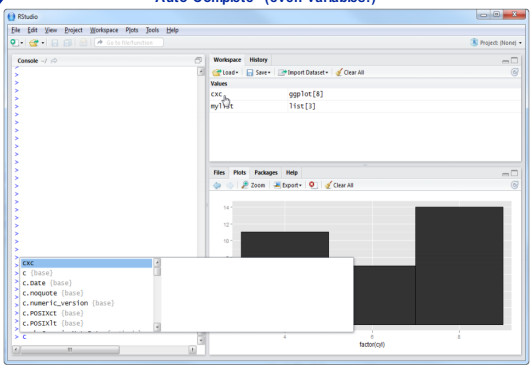
IDE Layout



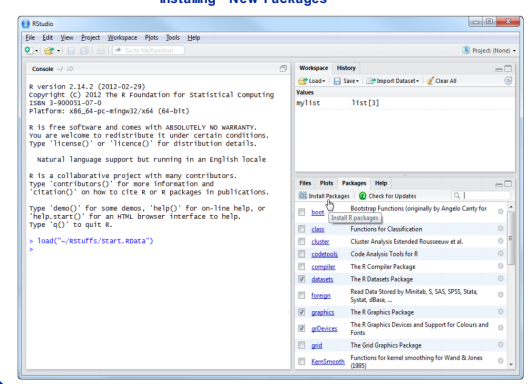
Auto-Complete (use tab)



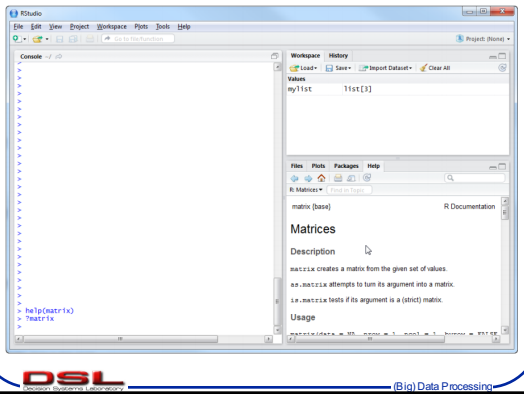
Auto-Complete (even variables!)



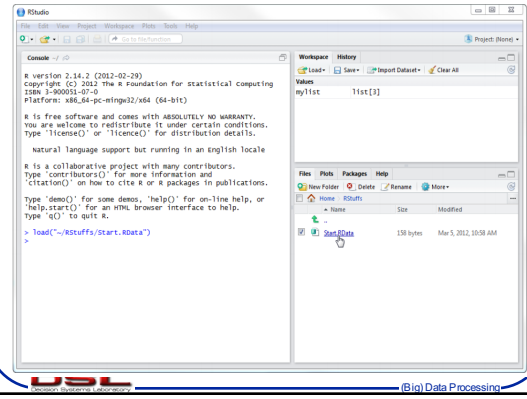
Installing New Packages



Need Help with functions? Use ? Or help()



Loading a file



R 101

In contrast to other Programming languages like C and java in R the **variables are not declared as some data type**.

The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable ***just like in Python***

There are many types of **R-objects**.

The frequently used ones are
- **Vectors**
- **Lists**
- **Matrices**
- **Arrays**
- **Factors**
- **Data Frames**

R - Data Types/Classes

Data Type	Example	Command Line
Logical	TRUE , FALSE	v <- TRUE print(class(v)) it produces following result: [1] "logical"
Numeric	12.3, 5, 999	v <- 23.5 print(class(v)) it produces following result: [1] "numeric"
Integer	2L, 34L, 0L	v <- 2L print(class(v)) it produces following result: [1] "integer"
Complex	3 + 2i	v <- 2+5i print(class(v)) it produces following result: [1] "complex"
Character	'a' , "good", "TRUE", '23.4'	v <- "TRUE" print(class(v)) it produces following result: [1] "character"
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <- charToRaw("Hello") print(class(v)) it produces following result: [1] "raw"

Variable Assignment

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using **print()** or **cat()**function. The **cat()** function combines multiple items into a continuous print output.

```
# Assignment using equal operator.
var1 = "abcdefg"

# Assignment using leftward operator.
var2 <- 12345

# Assignment using rightward operator.
c(TRUE,1) -> var.3 print(var.1) cat ("var.1
is ", var.1 ,"\n") cat ("var.2 is ", var.2
,"\n") cat ("var.3 is ", var.3 ,"\n")
```


Vectors

A **vector** is a sequence of data elements of the same basic type. Members in a vector are officially called **components**.

When you want to create vector with more than one element, you should use **c() concatenate function** which means to combine the elements into a vector.

```
# Create a vector.
apple <- c('red','green',"yellow")
>apple

 # Get the class of the vector.
 class(apple)
```

When we execute above code, it produces following result:

```
[1] "red" "green" "yellow"

[1] "character"
```


Creating Vectors

The `c()` function can be used to create vectors of objects.

```
> 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
```

Using the `vector()` function

```
> x <- vector("numeric", length = 10)
> x
 [1] 0 0 0 0 0 0 0 0 0 0
```

Let's Make a Vector!

- Remember me? I'm a Python scalar.

someInt = 12

someString = "hello Pittsburgh!"

- Remember me? I'm a Python list.

someIntList = [1,3,4,19]

someStringList = ["apple", "banana", "kittens"]

- In R, I am handled like this:

> newVector = c(1,2,4,19)

> newVector

[1] 1 2 4 19

> newVector[2]

[1] 2 (Wait a second!) (2 = 2?) (Shouldn't 1 = 2?)

Mixing Objects

What about the following?

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

When different objects are mixed in a vector, *coercion* occurs so that every element in the vector is of the same class.

1 is the new 0

- "A single 0 in an index position returns an empty structure;. x[0] returns named numeric(0)."
- So in Python it's 0.. In R, its 1
  - Got that? Good..

http://cran.r-project.org/doc/contrib/R_language.pdf

Matrices

All the elements of a matrix must be of the same type (numeric, logical, character, complex).

Matrices

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

```
> 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
```


Let's Add Some Labels!

```
>someMatrix = matrix(1:16, 4, 4, FALSE,
  dimnames=list(c("A","B","C","D"),c("W","X","Y","Z")))
> someMatrix
  W X Y Z
A 1 5 9 13
B 2 6 10 14
C 3 7 11 15
D 4 8 12 16
> someMatrix["B","Y"]
[1] 10
```

*Note: Use NULL if you don't want to append labels to either x or y axis


```
# Create a matrix. M = matrix( c('a','a','b','c','b','a'),
nrow=2,ncol=3,byrow = TRUE)
```

Matrix Example

[No Title]

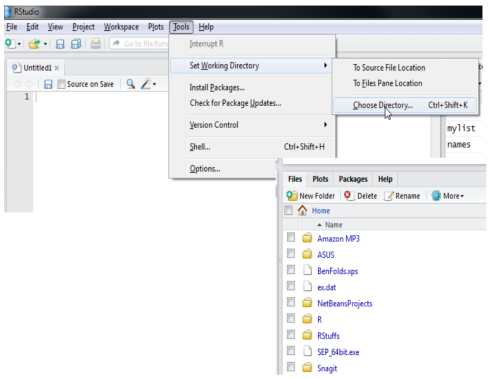
```
> someMatrix = matrix(1:16,4,4)
> someMatrix
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> someMatrix[10]
[1] 10
> someMatrix[2,3]
[1] 10
```

Data Frames

Data frames are tabular data objects. **Unlike a matrix in data frame each column can contain different modes of data**. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the **data.frame()** function.

```
# Create the data frame.
BMI <- data.frame(  gender = c("Male",  "Male","Female"),
height = c(152,  171.5,  165),  weight = c(81,93,  78),
Age =c(42,38,26)   )

print(BMI)
```

Read.Table()

- The majority of data is handled using a table (especially in ggplot)

USAGE: read.table(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", text)

Header

Made a slight change to "BeerAndWinePerCapita.txt"

State,GallonsOfBeer,GallonsOfWine
nEVADA,44,5.75

fromFile = read.table("BeerAndWinePerCapita.txt", header = TRUE, sep = ",")

> fromFile

	State	GallonsOfBeer	GallonsOfWine
1	nEVADA	44	5.75
2	nEW hAMPSHIRE	43.4	6.26
3	nORTH dAKOTA	41.7	1.56
4	mONTANA	41.5	3.06
5	sOUTH dAKOTA	39	1.50
6	wISCONSIN	38.2	2.63


```
>mydata <- read.table("Retention.txt",hea der=TRU E,s ep= "\t ")
>summary(mydata)
```

spend	apret	top10	rejr
Min. : 4125	Min. :18.75	Min. : 8.00	Min. : 0.00
1st Qu.: 7372	1st Qu.:45.37	1st Qu.:22.00	1st Qu.:19.17
Median : 9265	Median :55.71	Median :30.00	Median :27.39
Mean :10975	Mean :56.72	Mean :38.46	Mean :30.65
3rd Qu.:12838	3rd Qu.:68.69	3rd Qu.:49.50	3rd Qu.:36.81
Max. :35863	Max. :95.25	Max. :98.00	Max. :84.07
tstsc	pacc	strat	salar
Min. :48.12	Min. : 8.964	Min. : 7.20	Min. :38640
1st Qu.:61.11	1st Qu.:33.904	1st Qu.:13.40	1st Qu.:54650
Median :64.78	Median :40.850	Median :16.00	Median :61150
Mean :66.16	Mean :43.173	Mean :16.09	Mean :61358
3rd Qu.:70.45	3rd Qu.:51.773	3rd Qu.:18.57	3rd Qu.:67100
Max. :87.50	Max. :76.253	Max. :29.20	Max. :87900

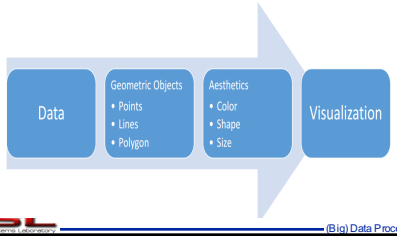
Factors

- **Nominal (Categorical) –**
  - Two or more categories
  - Have no intrinsic ordering
  - Think: (Male/Female), (Blonde/Brunette/Red Hair), and (Pittsburgh/State College/Erie/etc.)
- **Ordinal**
  - Similar to Nominal, but with order
  - Think: (Low/Medium/High), (Tall/Average/Short), and (Tall Latte/Grande Latte/Venti Latte no whip)
- **Interval**
  - Same as Ordinal, but evenly spaced
  - Think: (Temperature), (Time), and (Measurements)

ggplot2

Some Basics of ggplot2

- Extended library of R
- Used to create a variety of visualizations without a lot of background knowledge
- Created in a layered fashion



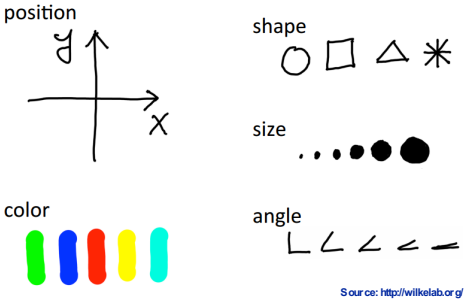
Traditonal plottng: You are a painter
– Manually place individual graphical elements

ggplot2: You employ a painter
– Describe conceptually how data should be visualized

Source: http://wilkelab.org/

Most confusing key concept: aesthetic mapping
Maps data values to visual elements of the plot

A few examples of aesthetics



Source: http://wilkelab.org/

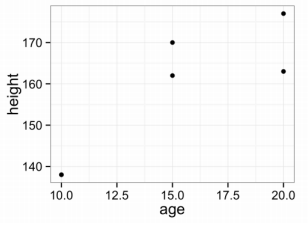
Simple example: mean height and weight of boys/girls ages 10-20

age (yrs)	height (cm)	weight (kg)	sex
10	138	32	M
15	170	56	M
20	177	71	M
10	138	33	F
15	162	52	F
20	163	53	F

Source: http://wilkelab.org/

Map age to x, height to y, visualize using points

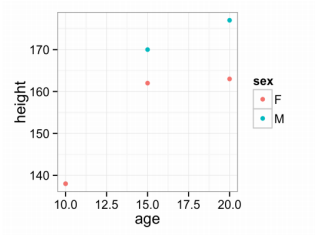
```
ggplot(data,  aes(x=age,  y=height))  + geom_point()
```



Source: http://wilkelab.org/

Let's color the points by sex

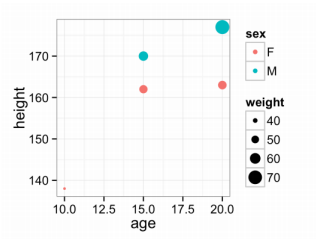
```
ggplot(data, aes(x=age, y=height,color=sex)) +geom_point()
```



Source: http://wilkelab.org/

And change point size by weight

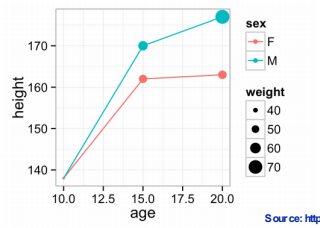
```
ggplot(data, aes(x=age, y=height,color=sex, size=weight)) +
geom_point()
```



Source: http://wilkelab.org/

The weight-to-size mapping should only be applied to points

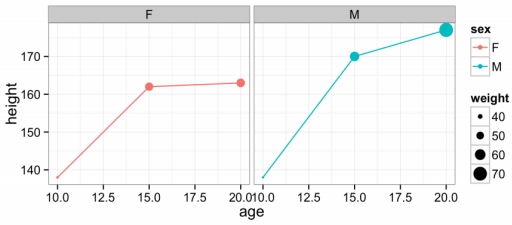
```
ggplot(data, aes(x=age, y=height,color=sex)) +
geom_point(aes(size=weight)) +
geom_line()
```



Source: http://wilkelab.org/

We can also make side-by-side plots (called facets)

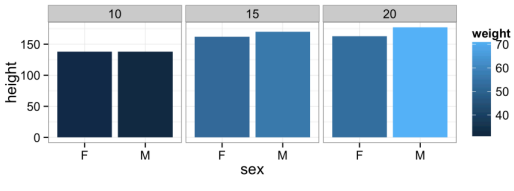
```
ggplot(data, aes(x=age, y=height,color=sex)) +
geom_point(aes(size=weight)) + geom_line() +
facet_wrap(~sex)
```



Source: http://wilkelab.org/

Now let's facet by age, color by weight, and use bars to plot height

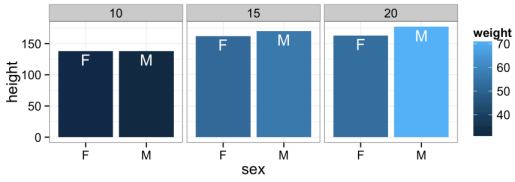
```
ggplot(data, aes(x=sex, y=height,fill=weight)) +
geom_bar(stat='identity') +facet_wrap(~age)
```



Source: http://wilkelab.org/

Let's plot the sex also at the top of the bar

```
ggplot(data, aes(x=sex, y=height, fill=weight)) +
geom_bar(stat='identity') +
geom_text(aes(label=sex), vjust=1.3, color='white') +
facet_wrap(~age)
```



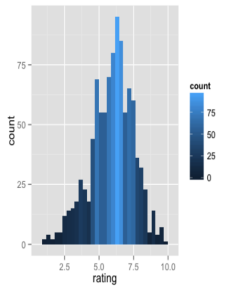
Source: http://wilkelab.org/

All the geom's with all their options are described on the ggplot2 web page

http://docs.ggplot2.org/current/

More Examples

Histogram



- http://docs.ggplot2.org/current/geom_histogram.html

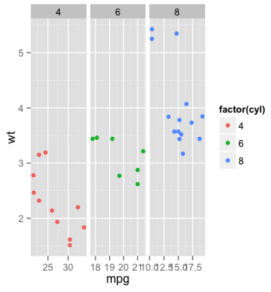
Facet Grid

ggplot2 Quick Reference: **facet**

The faceting approach supported by ggplot2 partitions a plot into a matrix of panels. Each panel shows a different subset of the data. There are two faceting approaches:

- **facet_wrap**(~cell) - univariate: create a 1-d strip of panels, based on one factor, and wrap the strip into a 2-d matrix
- **facet_grid**(row~col) - (usually) bivariate: create a 2-d matrix of panels, based on two factors

Facet Grid

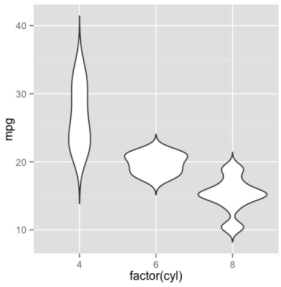


- http://docs.ggplot2.org/current/facet_grid.html
- http://www.cookbook-r.com/Graphs/Facets_(ggplot2)/

Violin Plots

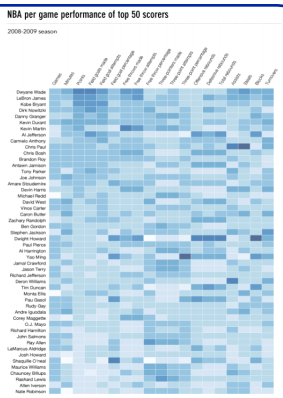
The violin plot is similar to box plots, except that they also show the probability density of the data at different values (in the simplest case this could be a histogram). Typically violin plots will include a marker for the median of the data and a box indicating the interquartile range, as in standard box plots. Overlaid on this box plot is a kernel density estimation.

Violin Plots



- http://docs.ggplot2.org/current/geom_violin.html

Heat Maps



- https://learnr.wordpress.com/2010/01/26/ggplot2-quick-heatmap-plotting/

Heat Maps cont…

https://learnr.wordpress.com/2010/01/26/ggplot2-quick-heatmap-plotting/

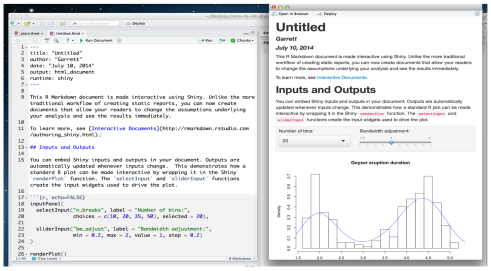
```
nba <- read.csv("http://datasets.flowingdata.com/ppg2008.csv")

nba$Name <- with(nba, reorder(Name, PTS))
```

With

- "Evaluate an R expression in an environment constructed from data, possibly modifying the original data."
- `nba$Name <- with(nba, reorder(Name, PTS))`

R Markdown

R Markdown is an authoring format that enables easy creation of dynamic documents, presentations, and reports from R.



- http://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf

Useful Links

- https://cran.r-project.org/

- http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/
- http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html
- http://blog.echen.me/2012/01/17/quick-introduction-to-ggplot2/
- http://had.co.nz/ggplot2/
- https://github.com/jennybc/ggplot2-tutorial
- http://www.ceb-institute.org.bbs/wp-content/uploads/2011/09/handout_ggplot2.pdf
- http://www.r-bloggers.com/basic-introduction-to-ggplot2/

Facet

http://sape.inf.usi.ch/quick-reference/ggplot2

Whisker

http://asq.org/learn-about-quality/data-collection-analysis-tools/overview/box-whisker-plot.html

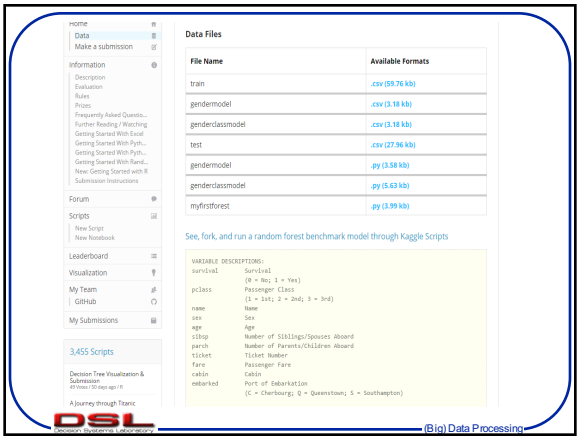
Violin

http://www.sthda.com/english/wiki/ggplot2-violin-plot-quick-start-guide-r-software-and-data-visualization

Machine learning kaggle titanic

https://www.datacamp.com/courses/kaggle-tutorial-on-machine-learning-the-sinking-of-the-titanic

Kaggle Titanic Assignment



Titanic Data Graphics

(1) Go to kaggle.com > Titanic: Machine Learning from Disaster and download (train.csv)

(2) Generate Descriptive Statistics

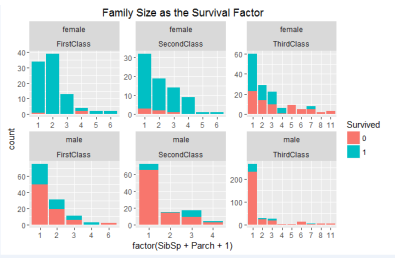
(3) You will create a plot (for each, so 5 plots in total) in ggplot2 using:

- **a. Whisker-plot**
- **b. Histogram**
- **c. Facet grid**
- **d. Violin plot**

(4) Write up a 500 – 1500 word document talking about the assignment using the graphics to describe the passengers of Titanic.

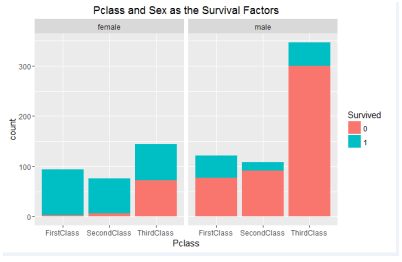
Titantic Examples


```
ggplot(total, aes(factor(SibSp + Parch + 1))) + + geom_bar(aes(fill = Survived)) + +
facet_wrap(~Sex+Pclass, nrow = 2, scales = "free") + + ggtitle("Family Size as the Survival Factor")
```



Source: Kaggle scripts


```
ggplot(total, aes(Pclass)) + + geom_bar(aes(fill = Survived)) + + facet_grid(~Sex) + +
ggtitle("Pclass and Sex as the Survival Factors")
```



Source: Kaggle scripts