

Lab: Learning R

Genomic Technologies Workshop 2022 (PLPTH885)

Sanzhen Liu

6/8/2024

Outlines

- Introduction of R programming
- Introduction of RStudio's IDE
- Running basic R
 - Data structure (vector and data frame)
 - Data importing and exporting
 - Plotting
 - String operations

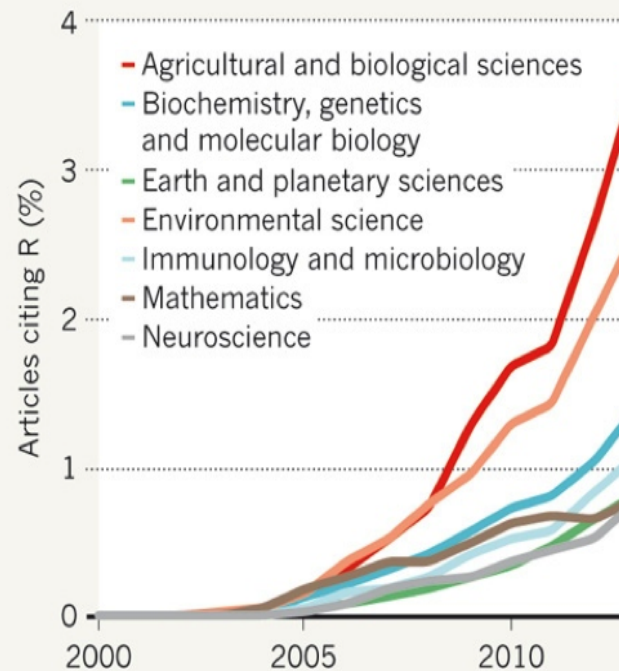
Why R?

- R is great at statistical computing and graphics
- R is a free software
- R has great community supports ([CRAN repository](#), Bioconductor, R-Forge, Github)

r-project.org

A RISING TIDE OF R

An increasing proportion of research articles explicitly reference R or an R package.



R example 1 (R for statistics)

Chi-square test

```
d <- c(12, 36, 24, 70)
dm <- matrix(d, nrow=2, byrow=T)
dm
```

```
      [,1] [,2]
[1,]   12  36
[2,]   24  70
```

```
chisq.test(dm)
```

Pearson's Chi-squared test with Yates' continuity correction

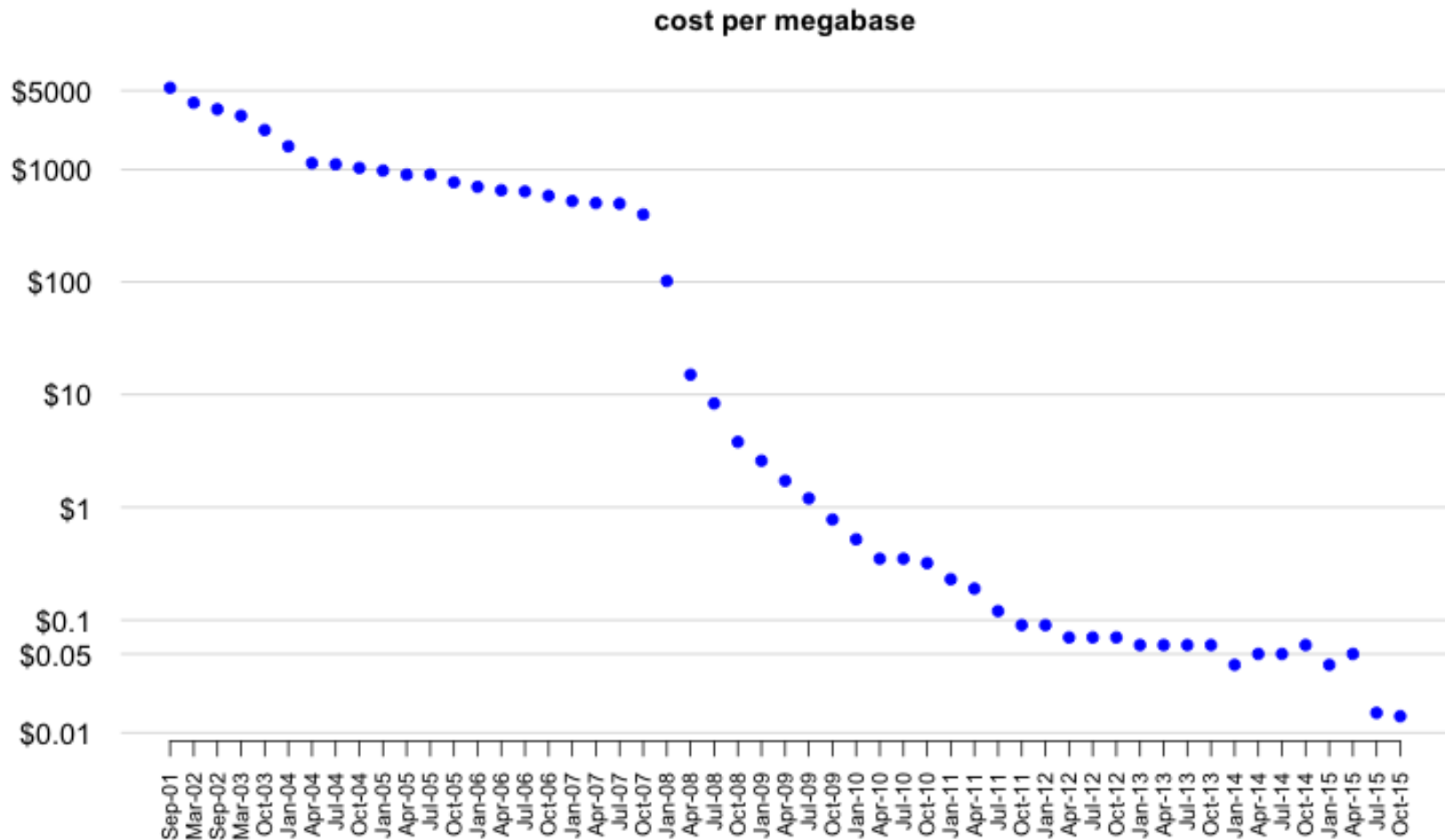
```
data:  dm
X-squared = 7.8894e-31, df = 1, p-value = 1
```

R example 2 (R for graphs)

```
ts="https://raw.githubusercontent.com/liu3zhenlab/teaching/master/RN/  
source(ts)
```

R example 3 (R for graphs)

```
cs="https://raw.githubusercontent.com/glin0311/teaching/master/RNA-Seq  
source(cs)
```

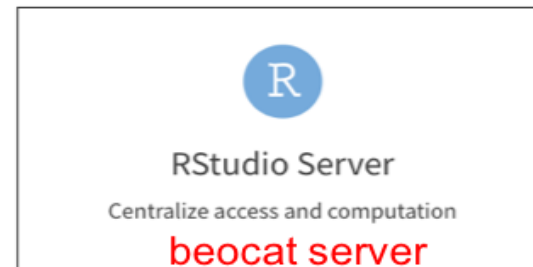
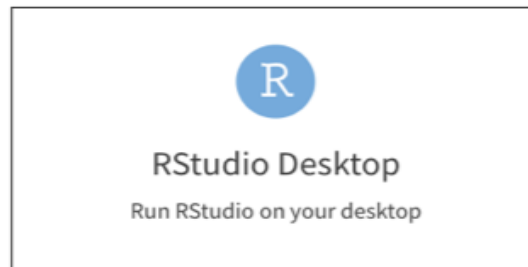


How to run R?

Rstudio is an open source integrated development environment (IDE) for R.

<https://www.rstudio.com/products/rstudio/>

There are two versions of RStudio:



Take a tour of RStudio's IDE

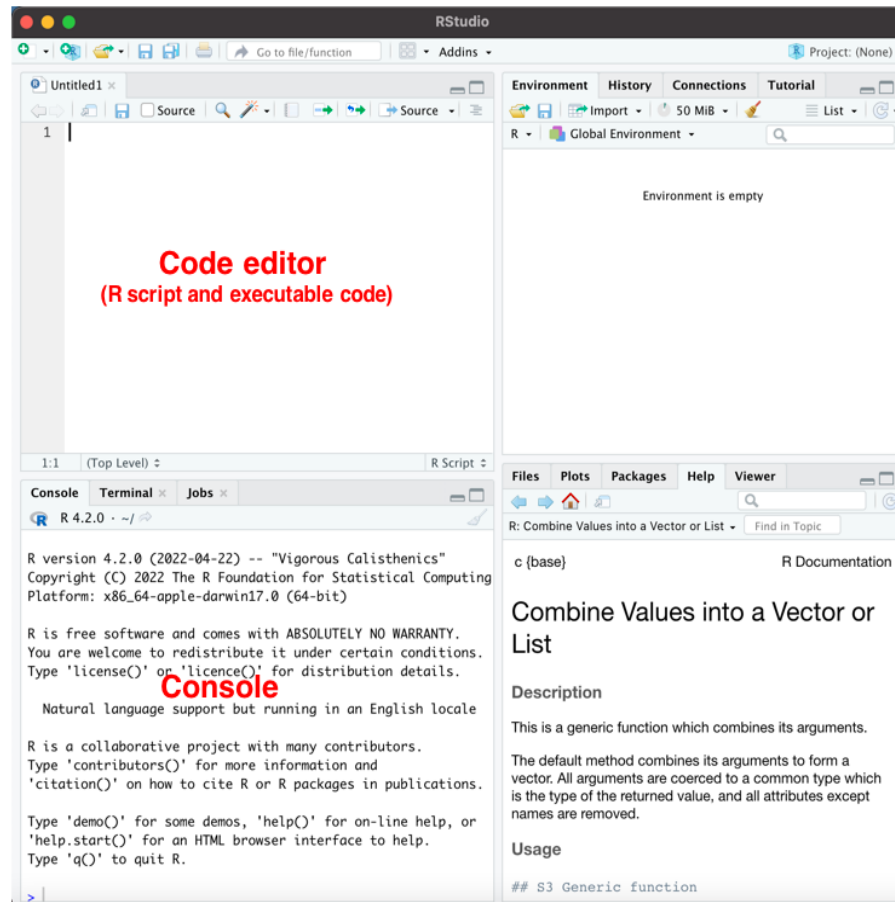


Install RStudio

- On your own machine (Rstudio Desktop)
 - Download and install **R**
 - Download and install **Rstudio**

RStudio Interface

RStudio IDE cheatsheet



- Executing commands for the code editor
 - PC window: control + return (enter)
 - Apple MAC: command + return (enter)

Setup working directory

Working directory is the default folder for input and output data.

```
dir.create("~/learnR") # create a directory for this practice.  
setwd("~/learnR") # setup working directory.
```

Install package

R packages contains many functions developed R community.

```
install.packages("learnr")  
library("learnr")
```

Getting started, R commands

Expression: evaluated, printed, and the value lost

```
2 + 4
```

```
[1] 6
```

```
68 * 0.15
```

```
[1] 10.2
```

Assignment

assign values to a variable
the value passed to a variable but NOT printed

assignment operator: <- or =

```
y <- 2  
y = 2  
y
```

```
[1] 2
```

```
info <- "hello world"  
cat(info)
```

```
hello world
```

Notes

- Comments (#): Notes to scripts, starting with a hashtag ('#'), everything to the end of the line is a comment.

```
y <- 2 + 4 # an example of the assignment
```

- Variable names are case sensitive

```
y <- 2  
Y <- 3  
y
```

```
[1] 2
```

```
Y
```

```
[1] 3
```

vector: multiple elements

A vector is a single entity consisting of an ordered collection of numbers, characters, logical quantities, etc.

concatenate command: **c()**

- Numeric vector
c(10.4, 5.6, 3.1, 6.4, 21.7)

vector manipulation (I)

```
# Numeric vector  
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)  
sum(x)
```

```
[1] 47.2
```

```
2*x
```

```
[1] 20.8 11.2  6.2 12.8 43.4
```

```
### extract 2nd elements  
x[2]
```

```
[1] 5.6
```


vector manipulation (II)

Logical vector

```
lv <- c(TRUE, FALSE, TRUE, TRUE)
!lv
```

```
## [1] FALSE TRUE FALSE FALSE
```

```
lv == FALSE
```

```
## [1] FALSE TRUE FALSE FALSE
```

vector manipulation (III)

```
# Character vector  
cv <- c("a", "b", "c")  
cv2 <- paste(cv, 1:3, sep="")  
cv2
```

```
[1] "a1" "b2" "c3"
```

```
# Missing value (NA, not available)  
mvv <- c("a", "b", "c", NA)  
is.na(mvv)
```

```
[1] FALSE FALSE FALSE  TRUE
```

vector manipulation (IV)

Vectors must have their values with the same mode, either numeric, character, logical, or other types. **conversion to other modes**

```
z <- 0:9  
is.numeric(z)
```

```
[1] TRUE
```

```
z
```

```
[1] 0 1 2 3 4 5 6 7 8 9
```

```
digits <- as.character(z) # convert to character  
digits
```

```
[1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

```
d <- as.numeric(digits) # convert to numeric  
d
```

```
[1] 0 1 2 3 4 5 6 7 8 9
```

vector manipulation (V)

- Select a subset of a vector

```
x <- c(4, 5, 7, 3, 9)
x[c(2, 3)]
```

```
[1] 5 7
```

```
x[x>6]
```

```
[1] 7 9
```

```
x[-c(1, 5)]
```

```
[1] 5 7 3
```

- Modify a vector

```
x[3] <- 23.1
c(x, 10.9)
```

```
[1] 4.0 5.0 23.1 3.0 9.0 10.9
```

Question

Can a vector contain different types of elements?

```
c(1, "a")  
c(1, TRUE)  
c(TRUE, "a")  
c(1, "a", TRUE)
```

Data frame

A data frame may be regarded as a matrix (table) with columns possibly of differing modes

- Making data frames

```
df <- data.frame(name=c("Josh", "rose"), age=c(23, 35))  
df
```

	name	age
1	Josh	23
2	rose	35

Working with a data frame

```
df
```

```
  name age  
1 Josh  23  
2 rose  35
```

Trying these commands:

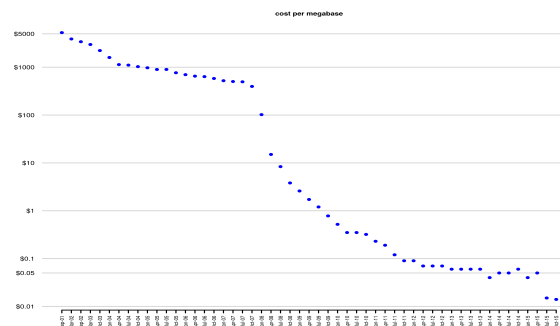
```
head(df, 1)  
tail(df, 1)  
str(df)  
df[2, 1]  
df[2, 2]  
df[2]  
df[, 2]
```

Importing data

read.table(): to read a data frame (table)
read.delim, read.csv

```
cpm="https://raw.githubusercontent.com/liu3zhenlab/teaching/master/RNA-Seq-Workshop/2022/data/cs.txt"
d <- read.delim(cpm)
head(d, 3)
```

	Date	Cost.per.Mb	Cost.per.Genome
1	Sep-01	5292.39	95263072
2	Mar-02	3898.64	70175437
3	Sep-02	3413.80	61448422



```
cpm="https://raw.githubusercontent.com/liu3zhenlab/teaching/master/RNA-Seq-Workshop/2022/data/cs.txt"
```


Exporting data

write.table() or **write.csv()**

To write a tab-delimited file

```
x <- data.frame(a = "pi", b = pi)
write.table(x, file="foo.txt", sep="\t", row.names=FALSE)
```

- file="foo.txt": foo.txt is the output file name
- sep="\t": separated by a tab (\t)
- row.names=FALSE: row names are not included in the output

Practice by your own

- Create a data frame

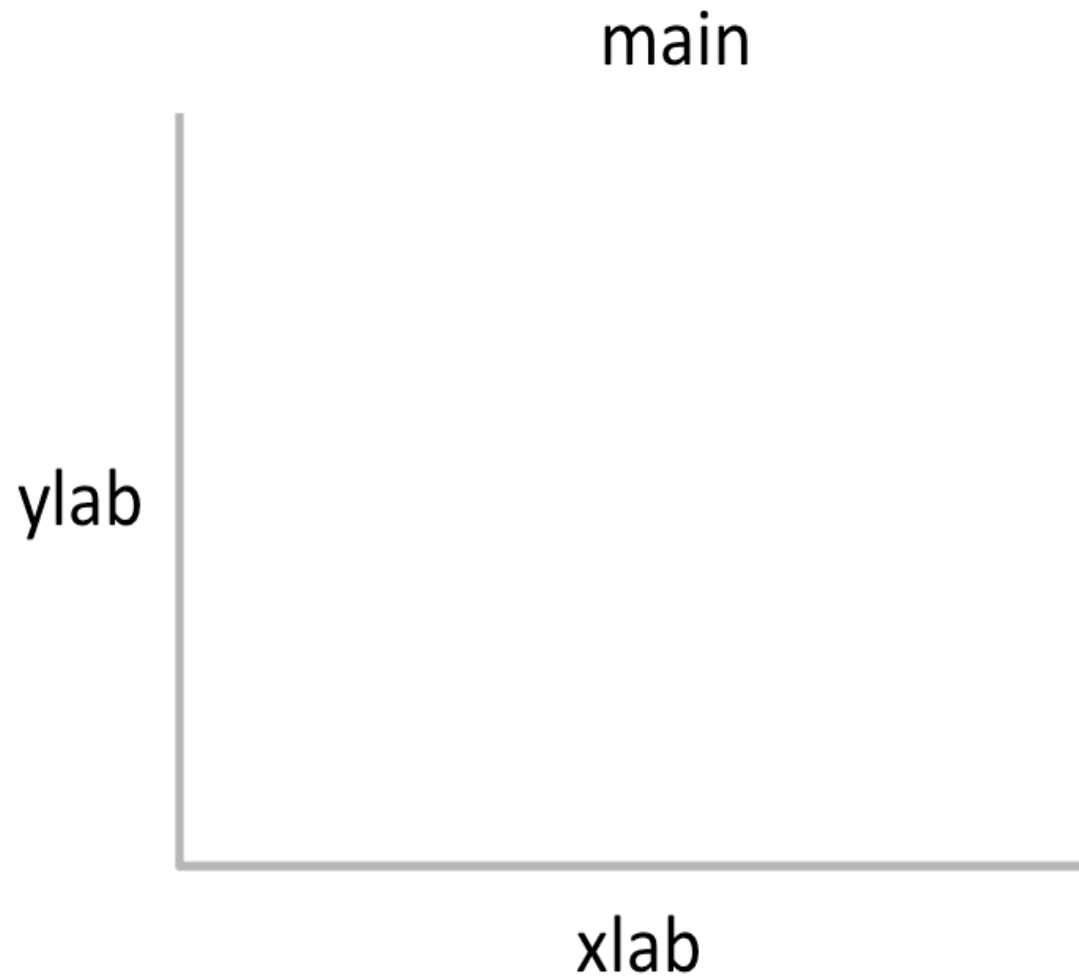
three columns: 1. Name 2. Major 3. Gender
three rows (entries): your neighbors and you

- Write the data frame to an output file
- Read the file to R and add one more column (e.g., favorite color)

Plotting: plot()

High-level plot: create a new plot

`plot(x, y, xlab, ylab, main, ...)`



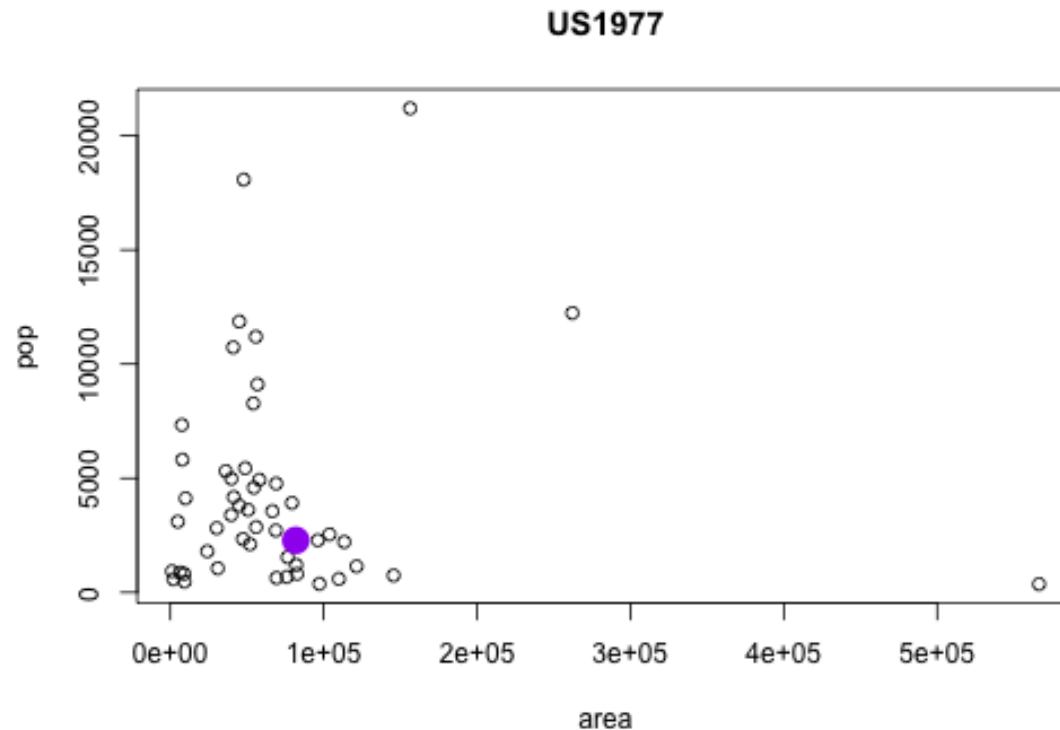
Adding contents to a plot

Low-level plot: add to an existing plot

- add points
points()
- add lines
lines()
- add text or legend
text()
legend()

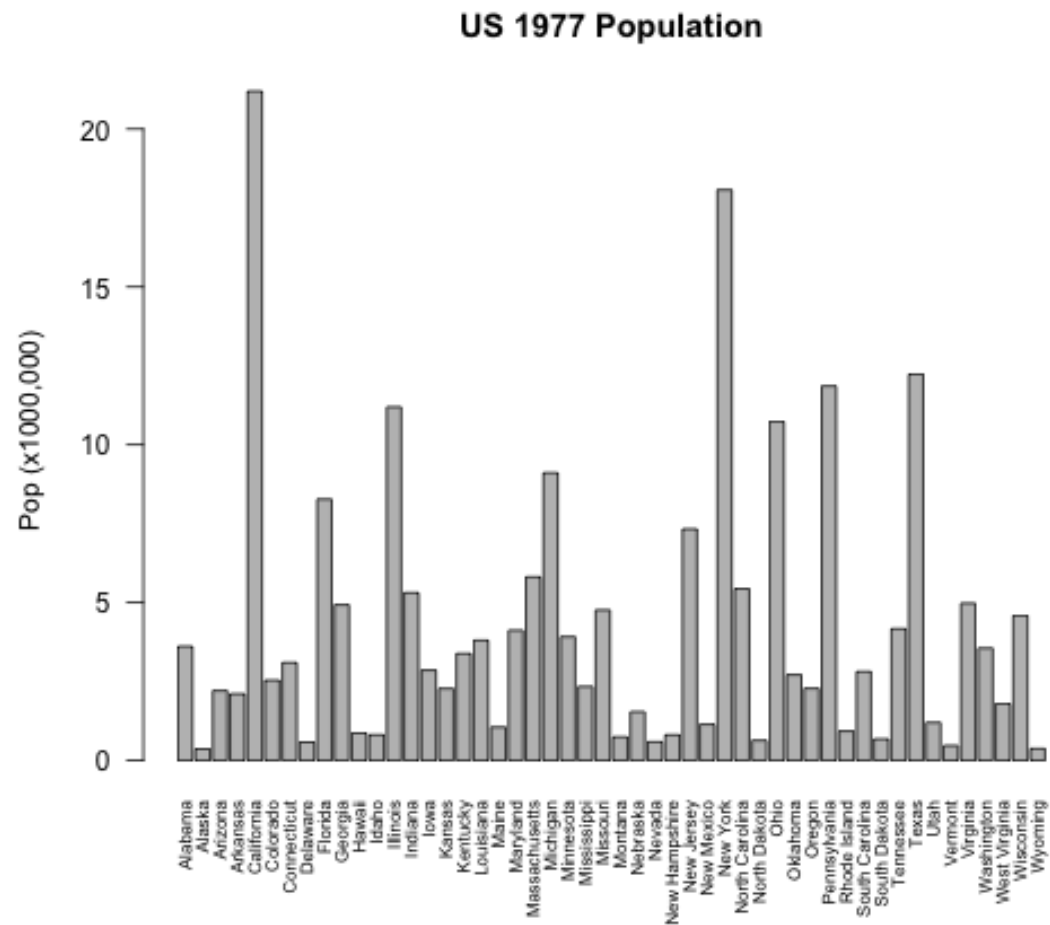
Scatter plot

```
area <- state.x77[, "Area"]
pop <- state.x77[, "Population"]
# scatter plot
plot(area, pop, main="US1977")
# label points
points(area["Kansas"], pop["Kansas"], col="purple",
       lwd=2, pch=19, cex=2)
```



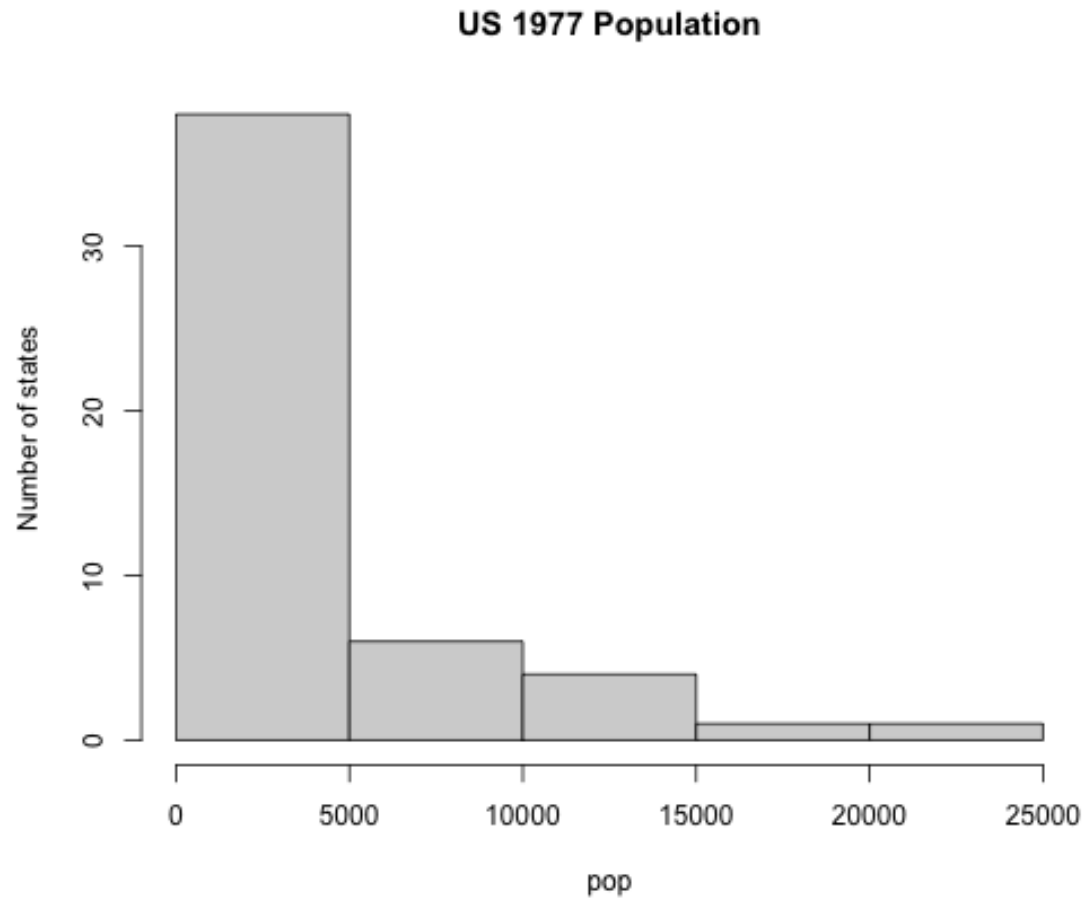
Boxplot

```
barplot(pop/1000, las=2, cex.names=0.65, ylab="Pop (x1000,000)",  
main="US 1977 Population")
```



Histogram

```
hist(pop, ylab="Number of states", main="US 1977 Population")
```



String operations - nchar

nchar() nchar the sizes of the corresponding elements of a vector.

```
cvec <- c("google", "hello", "the", "world")  
nchar(cvec)
```

```
[1] 6 5 3 5
```


String operations - grep

grep() grep searches for matches to argument pattern within each element of a character vector

```
cvec
```

```
[1] "google" "hello"  "the"    "world"
```

```
grep("o", cvec)
```

```
[1] 1 2 4
```

String operations – sub and gsub

sub() and **gsub()** sub and gsub perform replacement of the first and all matches respectively.

```
cvec
```

```
[1] "google" "hello"  "the"    "world"
```

```
sub("o", "O", cvec)
```

```
[1] "gOogle" "hellO"  "the"    "wOrld"
```

```
gsub("o", "O", cvec)
```

```
[1] "gOOgle" "hellO"  "the"    "wOrld"
```

Getting help

Usage of commands

- `help(nchar)`
- `?nchar`
- `??colsum`

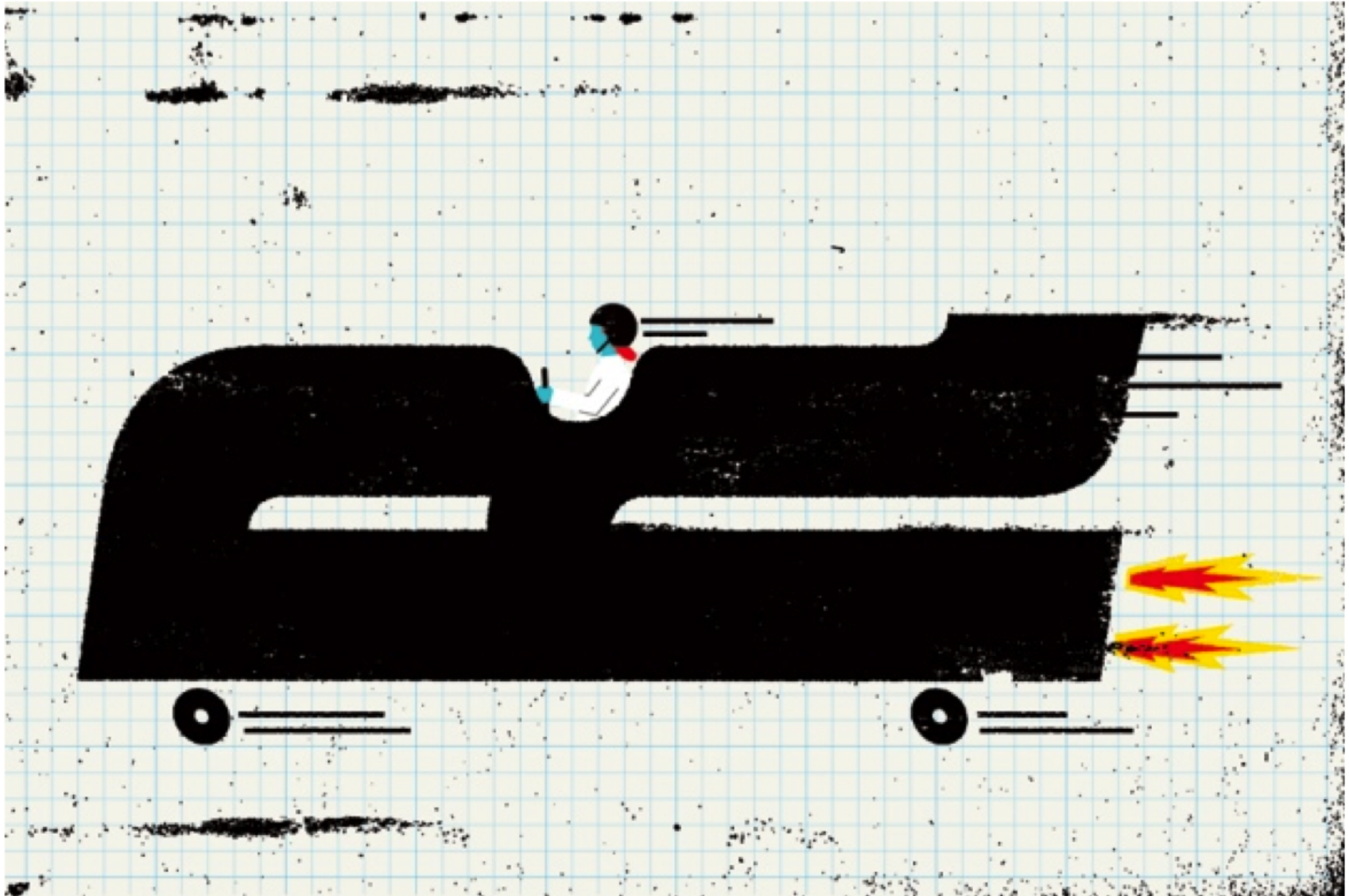
R Reference Card

- [stack overflow](#)
- [google](#)

Learning R at [swirlstats](#)

[education.rstudio.com](#)

Adventure with R



<http://www.nature.com/news/programming-tools-adventures-with-r-1.16609>