R TUTORIALS
for biologists

Weigang Qiu, Ph.D.
Department of Biological Sciences
Hunter College, City University of New York (CUNY)

Fall 2024
Comp Genomics @ KIZ
Workshop website:
https://wiki.genometracker.org/w/Computational_Genomics_(KIZ,_Fall_2024)
Qiu Lab wiki: https://wiki.genometracker.org/w/Main_Page

INSTALLATION

- Download and install *R* base/RStudio: https://posit.co/download/rstudio-desktop/
- Install, load & update “tidyverse”:
  - `install.packages(“tidyverse”)`
  - `library(“tidyverse”)`
  - `tidyverse_update()`

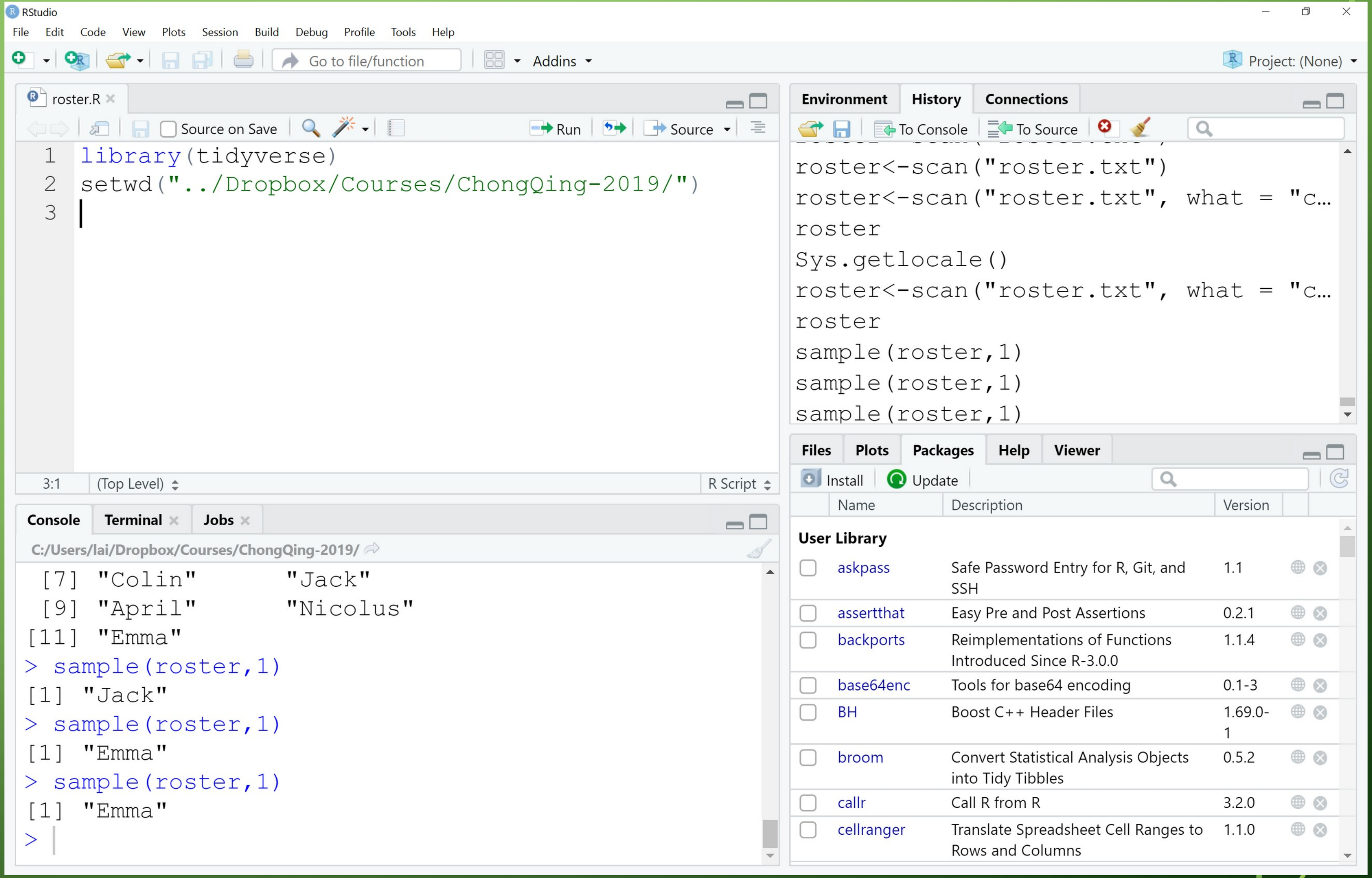
TUTORIAL 1
Getting Started

@QIU,
HUNTER/CUNY

Programming Terminology

- Windows/Panels
  - R Console
  - Script
  - History/Environment
  - Help/Plot/Packages
- Programming
  - Variable (cAsE SensItive!!)
  - Function
  - Argument
  - Object (data container)

@QIU, HUNTER/CUNY



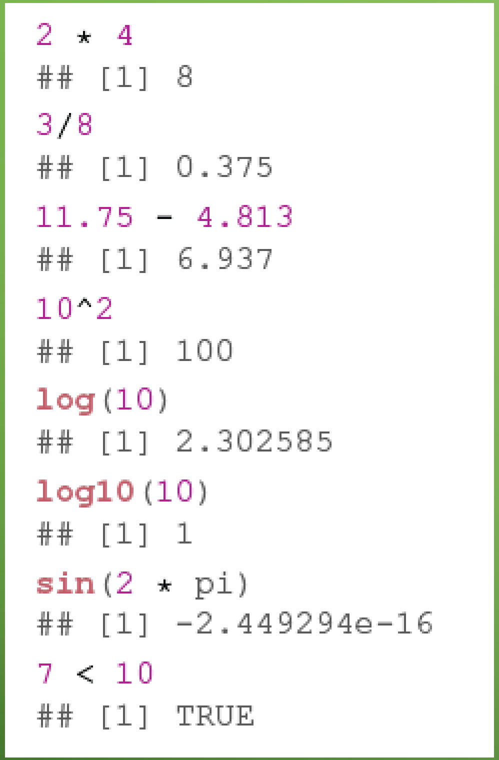
```
library(tidyverse)
setwd("../Dropbox/Courses/ChongQing-2019/")
```

```
C:/Users/lai/Dropbox/Courses/ChongQing-2019/
 [7] "Colin"       "Jack"
 [9] "April"       "Nicolus"
[11] "Emma"
> sample(roster,1)
[1] "Jack"
> sample(roster,1)
[1] "Emma"
> sample(roster,1)
[1] "Emma"
>
```

```
roster<-scan("roster.txt")
roster<-scan("roster.txt", what = "c…
roster
Sys.getlocale()
roster<-scan("roster.txt", what = "c…
roster
sample(roster,1)
sample(roster,1)
sample(roster,1)
```

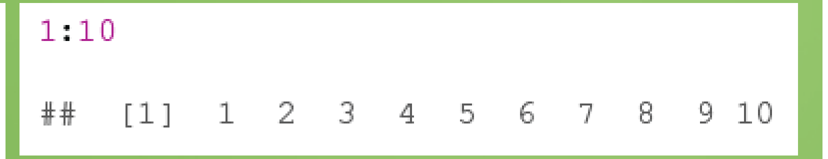
Name	Description	Version
User Library		
askpass	Safe Password Entry for R, Git, and SSH	1.1
assertthat	Easy Pre and Post Assertions	0.2.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.4
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.69.0-1
broom	Convert Statistical Analysis Objects into Tidy Tibbles	0.5.2
callr	Call R from R	3.2.0
cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0

Calculator



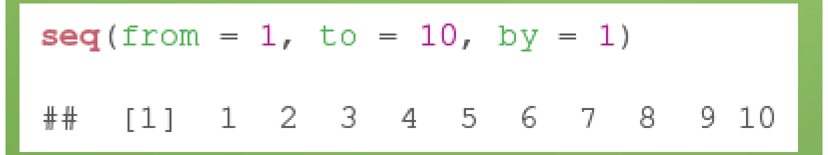
```
2 * 4
## [1] 8
3/8
## [1] 0.375
11.75 - 4.813
## [1] 6.937
10^2
## [1] 100
log(10)
## [1] 2.302585
log10(10)
## [1] 1
sin(2 * pi)
## [1] -2.449294e-16
7 < 10
## [1] TRUE
```

Vector



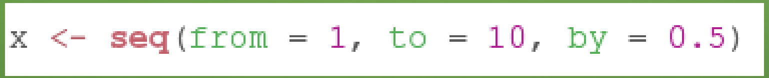
```
1:10

##  [1]  1  2  3  4  5  6  7  8  9 10
```

Function (w. arguments)



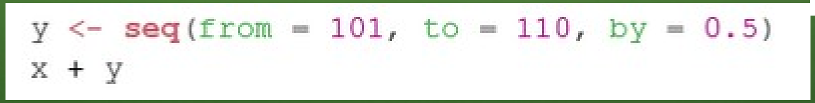
```
seq(from = 1, to = 10, by = 1)

##  [1]  1  2  3  4  5  6  7  8  9 10
```

Object (data container)



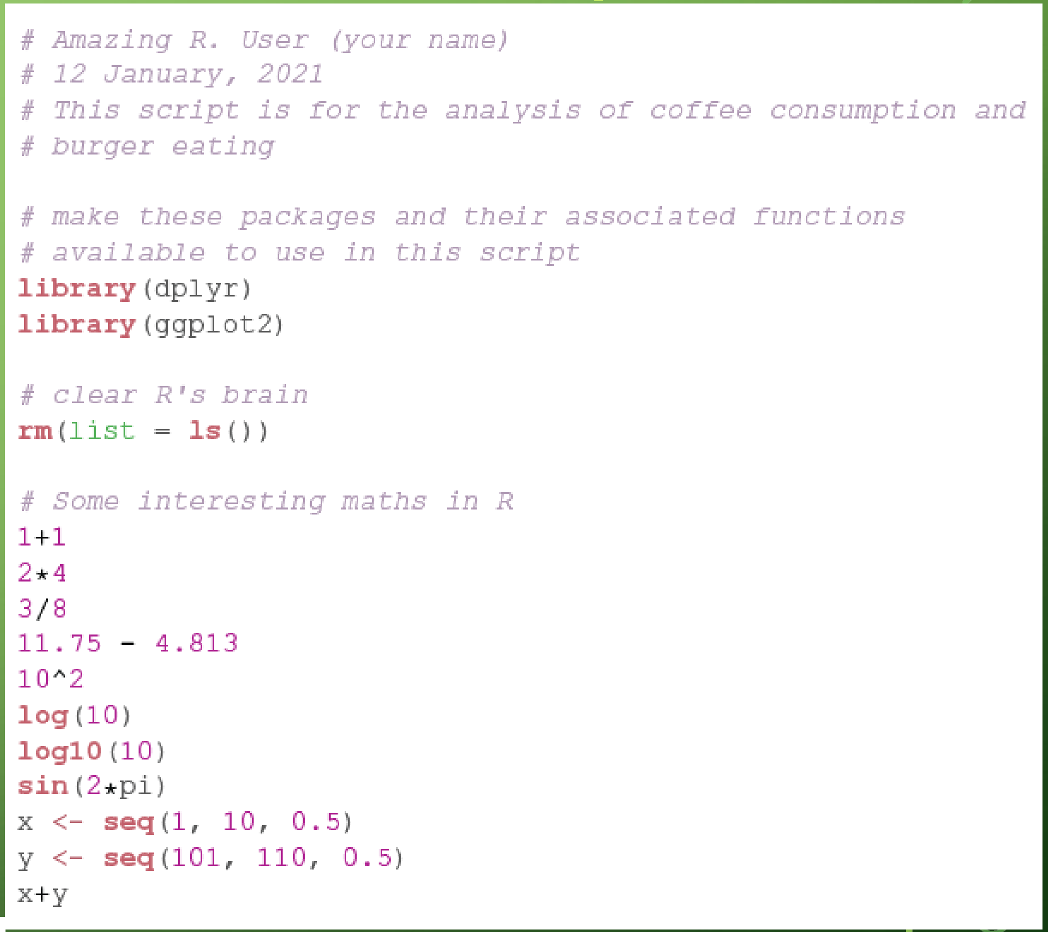
```
x <- seq(from = 1, to = 10, by = 0.5)
```

Vector operations



```
y <- seq(from = 101, to = 110, by = 0.5)
x + y
```

Save to script



```
# Amazing R. User (your name)
# 12 January, 2021
# This script is for the analysis of coffee consumption and
# burger eating

# make these packages and their associated functions
# available to use in this script
library(dplyr)
library(ggplot2)

# clear R's brain
rm(list = ls())

# Some interesting maths in R
1+1
2*4
3/8
11.75 - 4.813
10^2
log(10)
log10(10)
sin(2*pi)
x <- seq(1, 10, 0.5)
y <- seq(101, 110, 0.5)
x+y
```

To run: "ctl-enter"

Getting help: "?seq"

To save the output to an object: "<-"

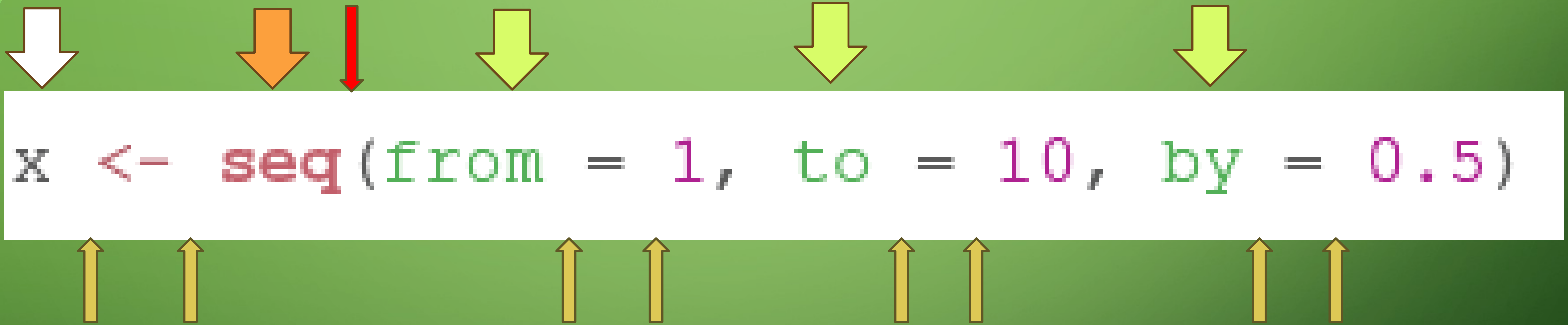
@QIU, HUNTER/CUNY

SUMMARY & TIPS

Object
Unquoted

Function()
Parenthesized
No space

Arguments (inputs)



```
x <- seq(from = 1, to = 10, by = 0.5)
```

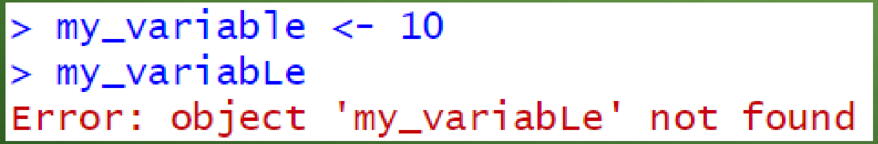
Add spaces around operators

A character vector (Quote the strings)

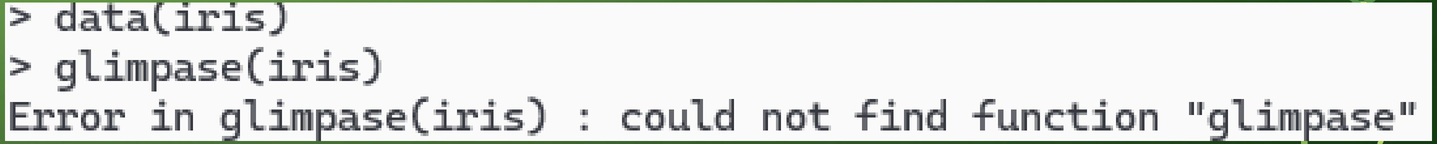
days.in.a.week ← c(“Mon”, “Tues”, “Wed”, “Th”, “Fri”, “Sat”, “Sun”)

PRACTICE #1

1. Create a variable called “my_first_variable” and assign it your last name
2. Create a vector containing the numbers 1, 3, 0.8, and 53.3, then assign it to a variable called “my_numbers”
3. Make a vector that contains all the numbers from 6 to 12, incremented by 0.5
4. Create and assign a vector that contains the numbers from 3 to 9. After assigning, square all the numbers
5. Create a vector of DNA bases; apply the function “sample()”. Explain the output. Look up help for this function. Create a random DNA sequence of 1000 bases
6. Why the following codes don’t work?



```
> my_variable <- 10
> my_variabLe
Error: object 'my_variabLe' not found
```



```
> data(iris)
> glimpase(iris)
Error in glimpase(iris) : could not find function "glimpase"
```

7. Save all commands in a file “practice-1.R”

TUTORIAL 2
Data Manipulation

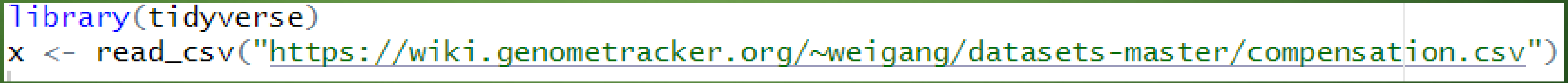
@QIU,
HUNTER/CUNY

LOAD DATA

1. Download the zip file containing data sets from http://www.r4all.org/the-book/datasets
2. Upload: Files ✉ Upload ✉ Choose file
3. Read a data file:

```
compensation  read_csv(“datasets-master/compensation.csv”)
```

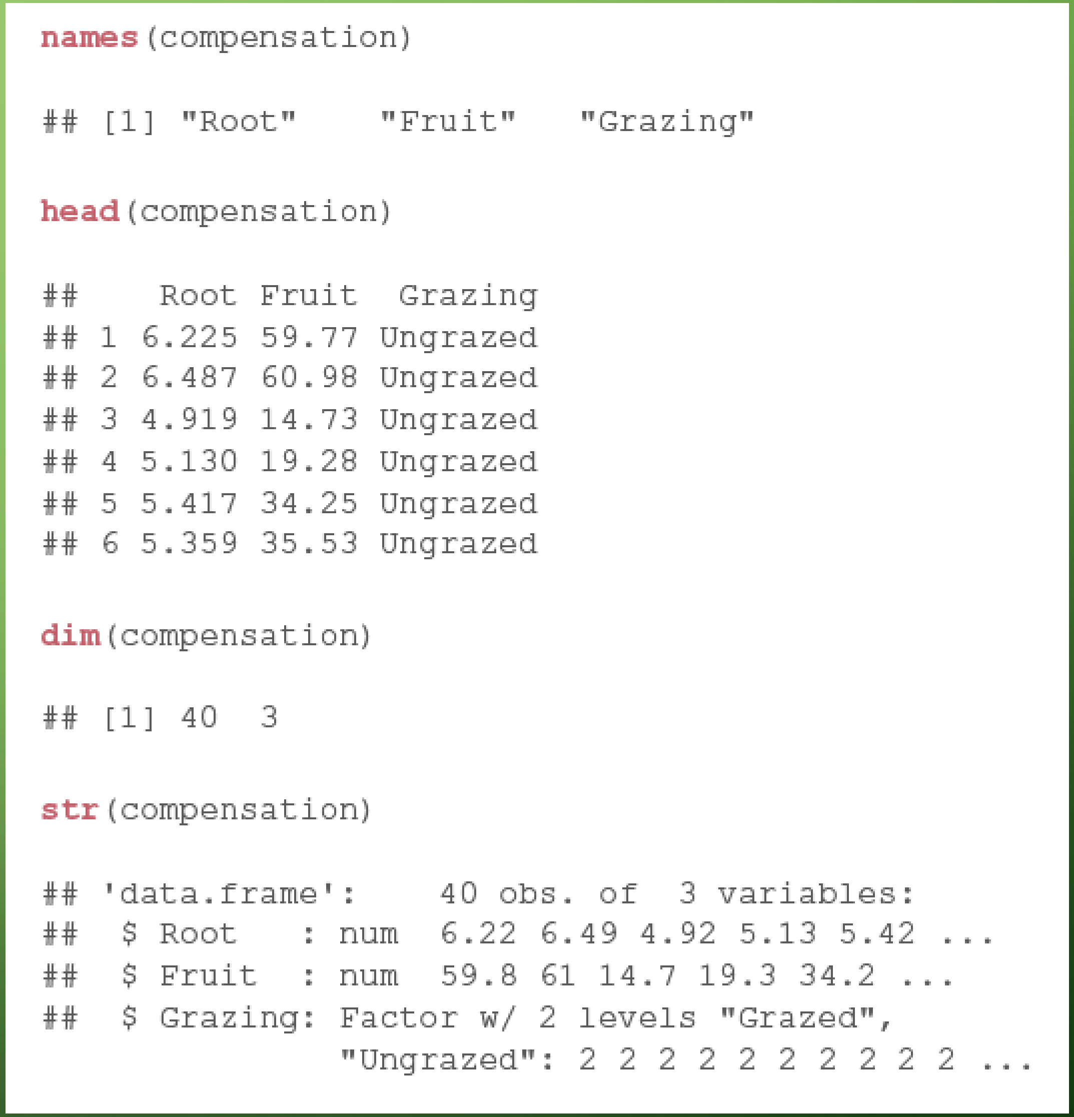
Alternatively, read from an online repository:



```
library(tidyverse)
x <- read_csv("https://wiki.genometracker.org/~weigang/datasets-master/compensation.csv")
```

THE “compensation” DATASET

1. Numeric variable “Fruit”: production of apple (in kg)
2. Numeric variable “Root”: width of rootstock (in cm)
3. Categorical variable “Grazing”: allowing for cattle grazing or not



```
names(compensation)

## [1] "Root"    "Fruit"   "Grazing"

head(compensation)

##    Root Fruit  Grazing
## 1 6.225 59.77 Ungrazed
## 2 6.487 60.98 Ungrazed
## 3 4.919 14.73 Ungrazed
## 4 5.130 19.28 Ungrazed
## 5 5.417 34.25 Ungrazed
## 6 5.359 35.53 Ungrazed

dim(compensation)

## [1] 40  3

str(compensation)

## 'data.frame':    40 obs. of  3 variables:
##  $ Root   : num  6.22 6.49 4.92 5.13 5.42 ...
##  $ Fruit  : num  59.8 61 14.7 19.3 34.2 ...
##  $ Grazing: Factor w/ 2 levels "Grazed",
             "Ungrazed": 2 2 2 2 2 2 2 2 2 2 ...
```

TIPS FOR VARIABLE & FILE NAMES

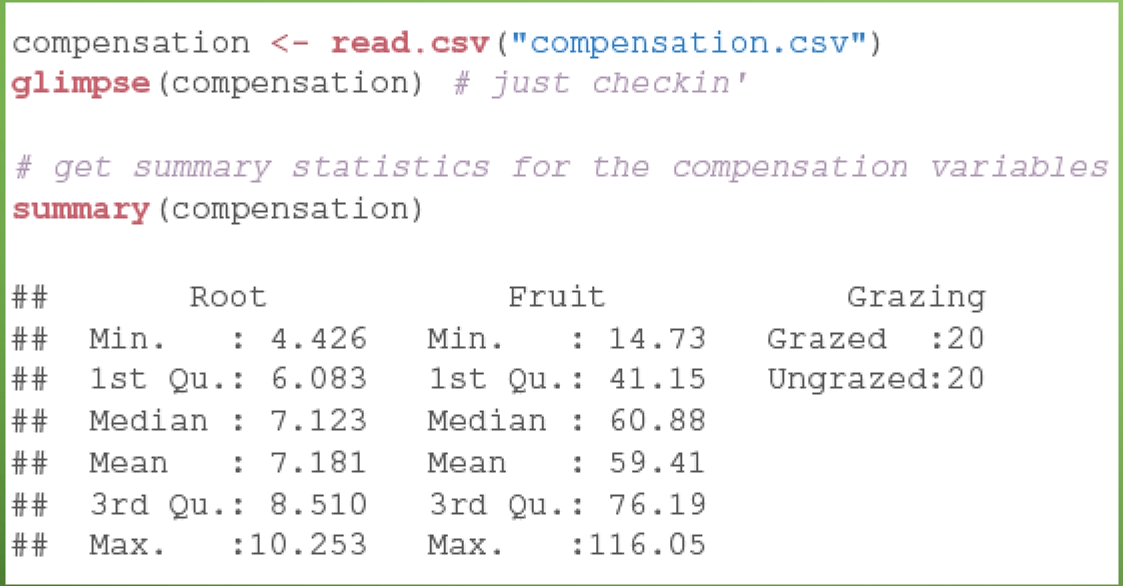
Computer-friendly variable/file names

- camelFormatName
- worm_format_name

Computer-unfriendly variable/file names

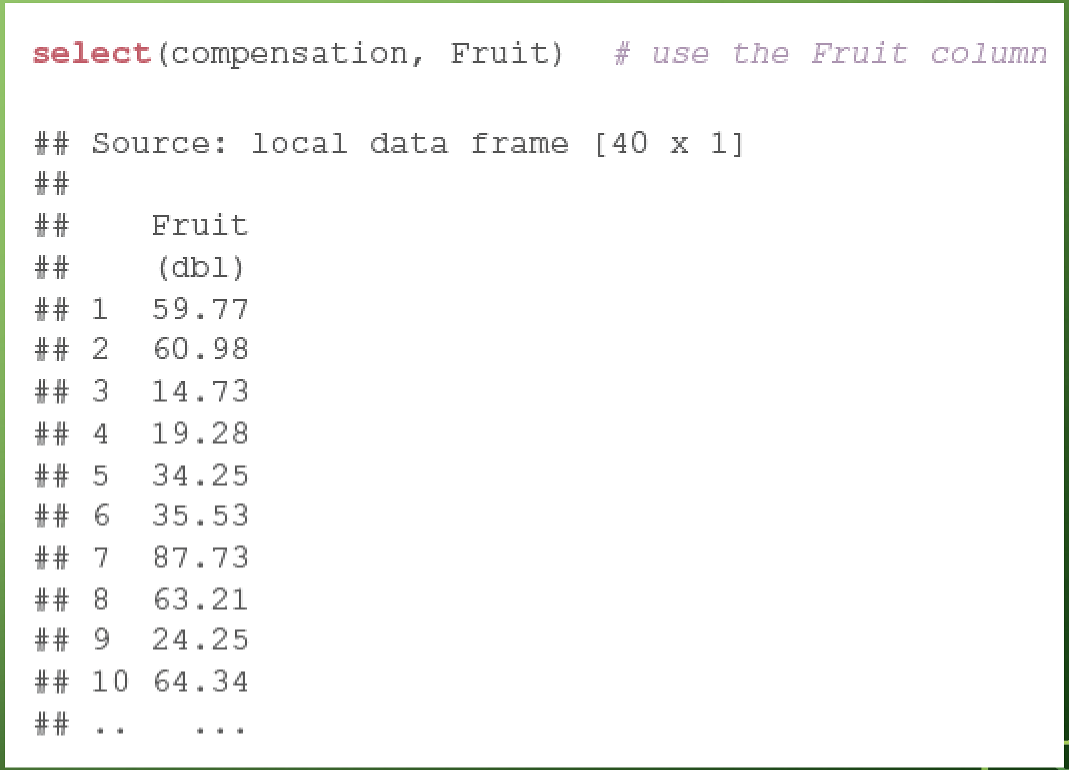
- “Name with spaces”
- “name-with-dashes”
- “123nameStartWithNumbers
- “a”, “b”, “c” (uninformative)

summary(): statistics



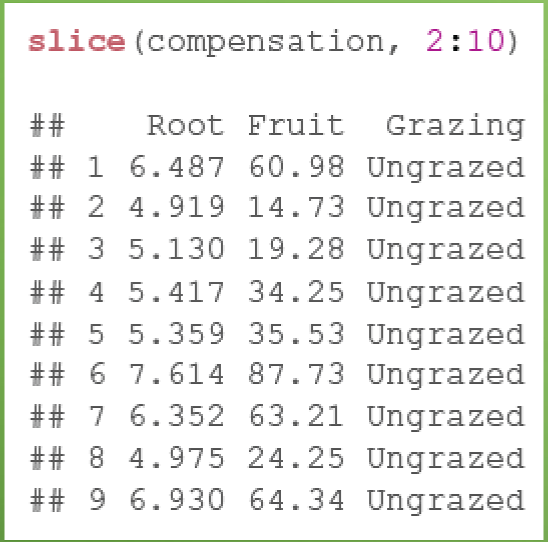
```
compensation <- read.csv("compensation.csv")
glimpse(compensation) # just checkin'

# get summary statistics for the compensation variables
summary(compensation)

##       Root             Fruit              Grazing
##  Min.   : 4.426   Min.   : 14.73   Grazed  :20
##  1st Qu.: 6.083   1st Qu.: 41.15   Ungrazed:20
##  Median : 7.123   Median : 60.88
##  Mean   : 7.181   Mean   : 59.41
##  3rd Qu.: 8.510   3rd Qu.: 76.19
##  Max.   :10.253   Max.   :116.05
```

select(): choose columns

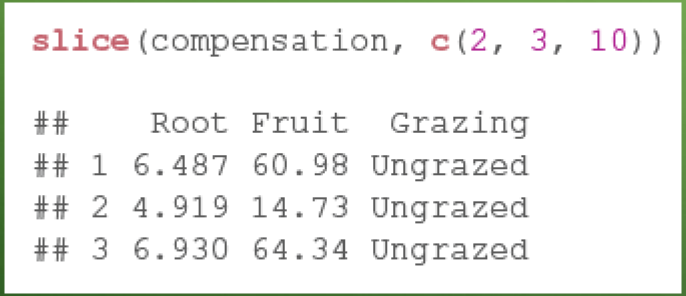


```
select(compensation, Fruit)  # use the Fruit column

## Source: local data frame [40 x 1]
##
##    Fruit
##    (dbl)
## 1  59.77
## 2  60.98
## 3  14.73
## 4  19.28
## 5  34.25
## 6  35.53
## 7  87.73
## 8  63.21
## 9  24.25
## 10 64.34
## ..   ...
```

slice(): choose rows

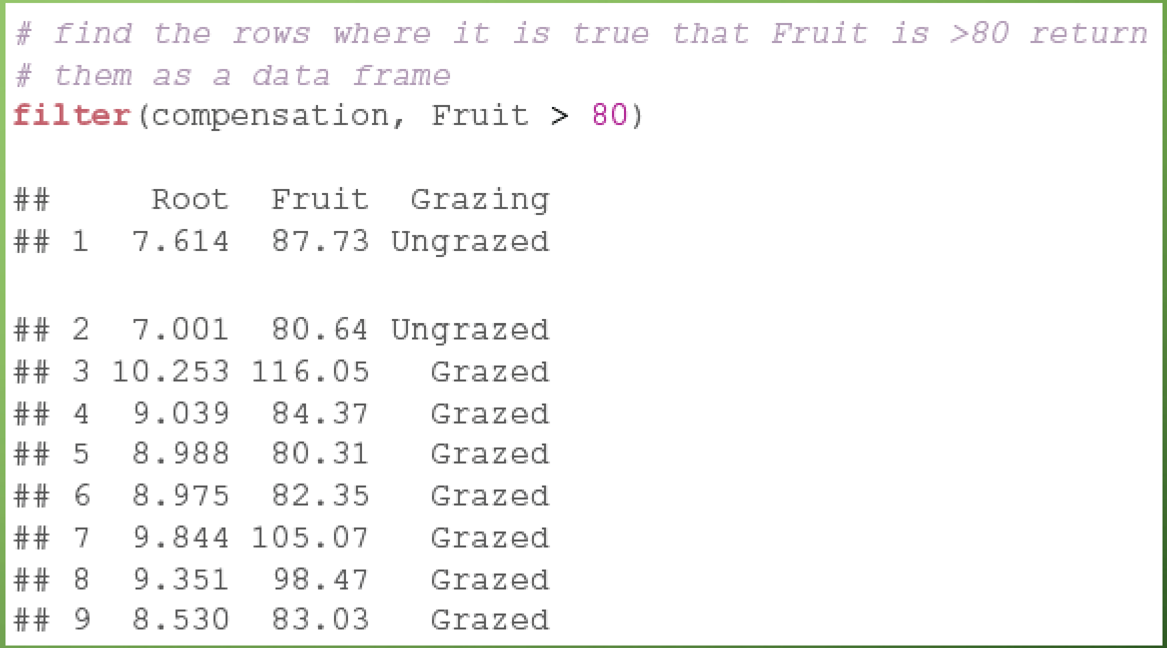


```
slice(compensation, 2:10)

##     Root Fruit  Grazing
## 1 6.487 60.98 Ungrazed
## 2 4.919 14.73 Ungrazed
## 3 5.130 19.28 Ungrazed
## 4 5.417 34.25 Ungrazed
## 5 5.359 35.53 Ungrazed
## 6 7.614 87.73 Ungrazed
## 7 6.352 63.21 Ungrazed
## 8 4.975 24.25 Ungrazed
## 9 6.930 64.34 Ungrazed
```

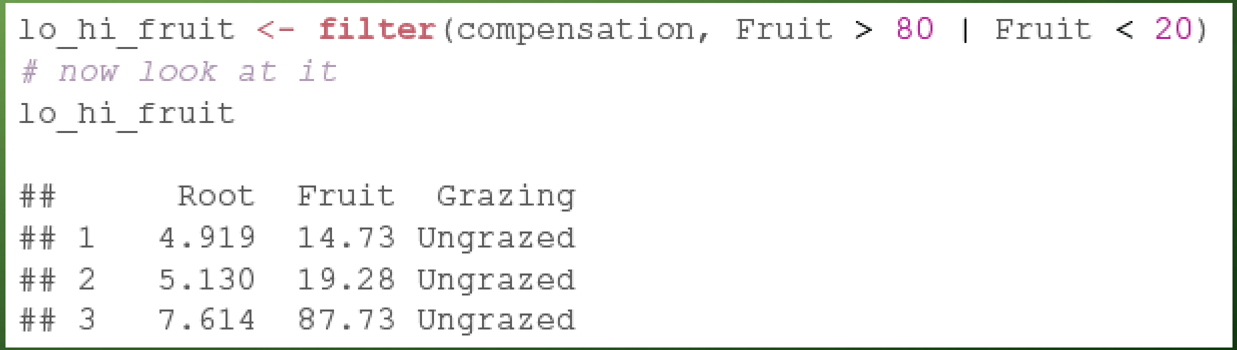


```
slice(compensation, c(2, 3, 10))

##     Root Fruit  Grazing
## 1 6.487 60.98 Ungrazed
## 2 4.919 14.73 Ungrazed
## 3 6.930 64.34 Ungrazed
```

filter(): conditional row filtering



```
# find the rows where it is true that Fruit is >80 return
# them as a data frame
filter(compensation, Fruit > 80)

##      Root  Fruit  Grazing
## 1  7.614  87.73 Ungrazed

## 2  7.001  80.64 Ungrazed
## 3 10.253 116.05   Grazed
## 4  9.039  84.37   Grazed
## 5  8.988  80.31   Grazed
## 6  8.975  82.35   Grazed
## 7  9.844 105.07   Grazed
## 8  9.351  98.47   Grazed
## 9  8.530  83.03   Grazed
```



```
lo_hi_fruit <- filter(compensation, Fruit > 80 | Fruit < 20)
# now look at it
lo_hi_fruit

##      Root  Fruit  Grazing
## 1   4.919  14.73 Ungrazed
## 2   5.130  19.28 Ungrazed
## 3   7.614  87.73 Ungrazed
```

mutate(): data transformation

```
# what does compensation look like now?
head(compensation)

##    Root Fruit  Grazing
## 1 6.225 59.77 Ungrazed
## 2 6.487 60.98 Ungrazed
## 3 4.919 14.73 Ungrazed
## 4 5.130 19.28 Ungrazed
## 5 5.417 34.25 Ungrazed
## 6 5.359 35.53 Ungrazed

# use mutate
# log(Fruit) is in the column logFruit
# all of which gets put into the object compensation
compensation <- mutate(compensation, logFruit = log(Fruit))

# first 6 rows of the new compensation
head(compensation)

##    Root Fruit  Grazing logFruit
## 1 6.225 59.77 Ungrazed 4.090504


## 2 6.487 60.98 Ungrazed 4.110546

## 3 4.919 14.73 Ungrazed 2.689886
## 4 5.130 19.28 Ungrazed 2.959068
## 5 5.417 34.25 Ungrazed 3.533687
## 6 5.359 35.53 Ungrazed 3.570377
```

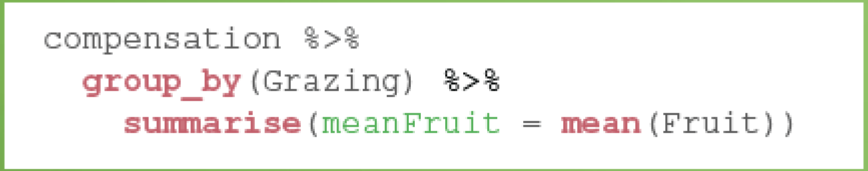
arrange(): sort rows

```
arrange(compensation, Fruit)

##    Root Fruit  Grazing logFruit
## 1 4.919 14.73 Ungrazed 2.689886
## 2 6.106 14.95   Grazed 2.704711
## 3 4.426 18.89 Ungrazed 2.938633
## 4 5.130 19.28 Ungrazed 2.959068
## 5 4.975 24.25 Ungrazed 3.188417
## 6 5.451 32.35 Ungrazed 3.476614
```

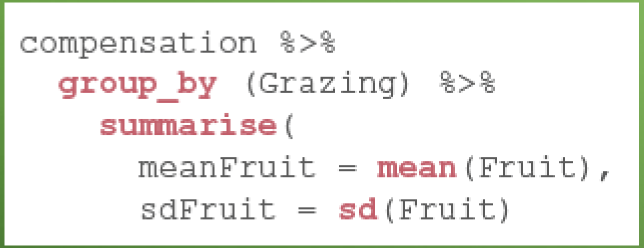
Chaining with "%>%" or " |>"

```
# Root values from Fruit > 80 subset
# Via piping
compensation %>%
  filter(Fruit > 80) %>%
    select(Root)

##      Root
## 1   7.614
## 2   7.001
## 3  10.253
## 4   9.039
## 5   8.988
## 6   8.975
## 7   9.844
## 8   9.351
## 9   8.530
```

Summarize by groups

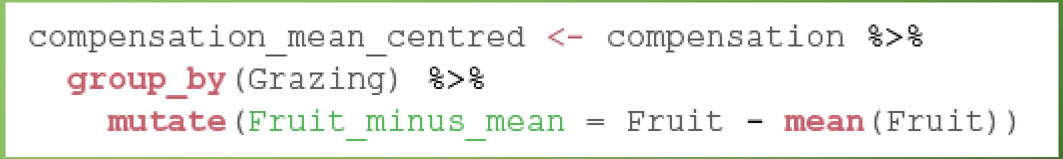


```
compensation %>%
  group_by(Grazing) %>%
    summarise(meanFruit = mean(Fruit))
```



```
compensation %>%
  group_by (Grazing) %>%
    summarise(
      meanFruit = mean(Fruit),
      sdFruit = sd(Fruit)
```

Transform by groups



```
compensation_mean_centred <- compensation %>%
  group_by(Grazing) %>%
    mutate(Fruit_minus_mean = Fruit - mean(Fruit))
```

PRACTICE #2

Load the "iris" dataset with `data("iris")` & answer the following questions:

1. Fine the dimensions of the dataset
2. List the variables and their data types (Hint: run `glimpse("iris")`)
3. Summarize the variables (Hint: run `summary("iris")`)
4. Get the last 10 observations of the dataset
5. Select only the first four columns (remove the "`Species`" column)
6. Filter rows by species, retain only rows from one species (e.g., "`setosa`")
7. Filter rows by a cutoff value (e.g., "`Sepal.Length >= 4`")
8. Add a column by taking the log10 of "`Sepal.Lengh`"
9. What are the medians of the variable "`Sepal.Length`" for each species?
10. Count how many samples for each species
11. Save all commands in a file "practice-2.R"