R for Genomics

Vasilis Lenis: vpl@aber.ac.uk

Michael Squance: mis20@aber.ac.uk

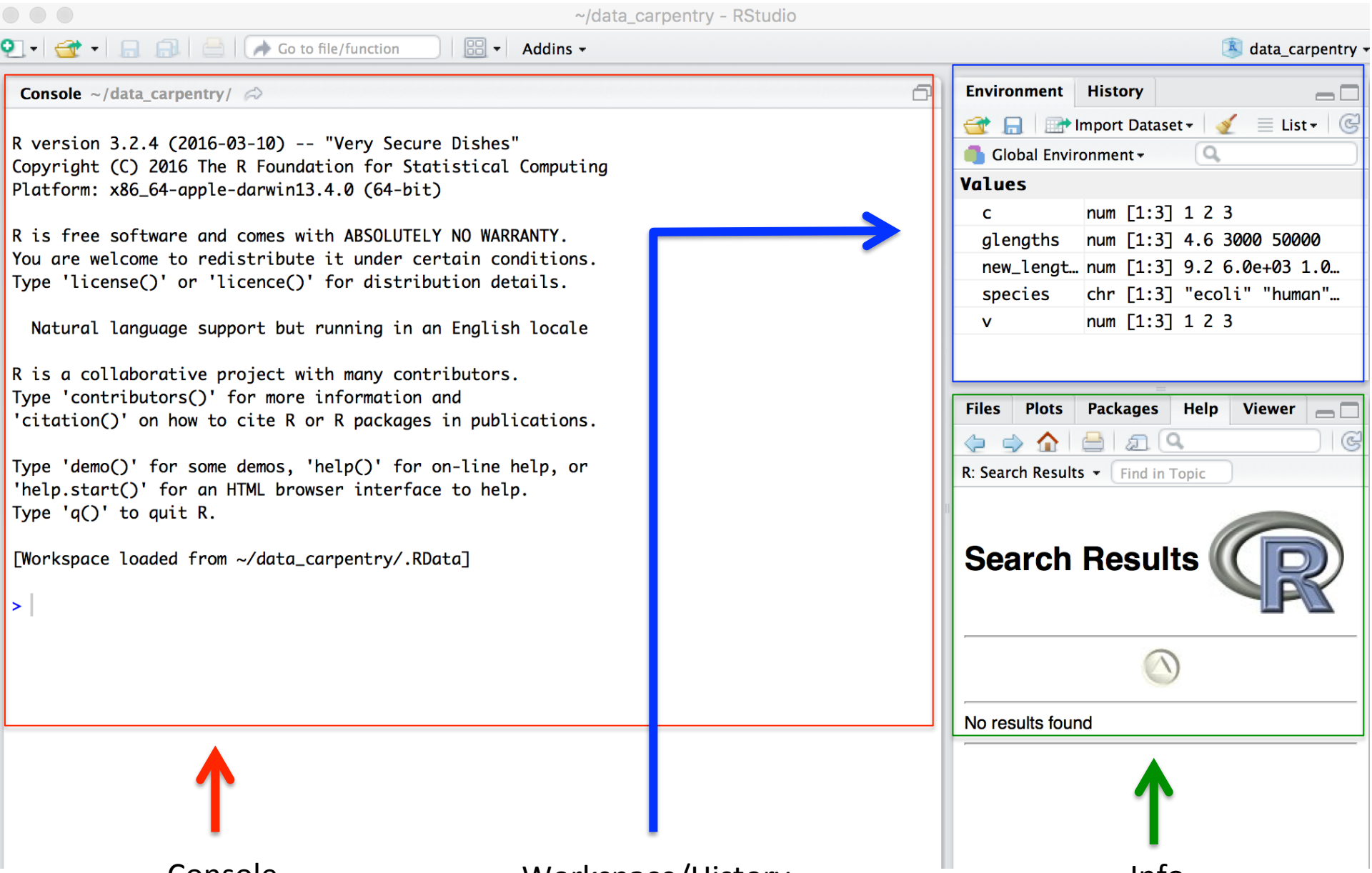
What is R?

- R is an open source programming/scripting language (Inspired by the programming language S).
- Useful for statistics and data science.
- Superior like commercial alternatives (over 7,000 user contributed packages at this time).
- Widely used both in academia and industry.
- Available on all platforms – general purpose programming.
- Large and growing community of peers.

Where and How

- How to get R:
  - http://www.r-project.org/
  - Google: “R”
  - Windows, Linux, Mac OS X, source

- Ways of interacting with R
  - Command line:
    - `user@vpl:~$> R`
  - GUI environment:
    - RStudio IDE



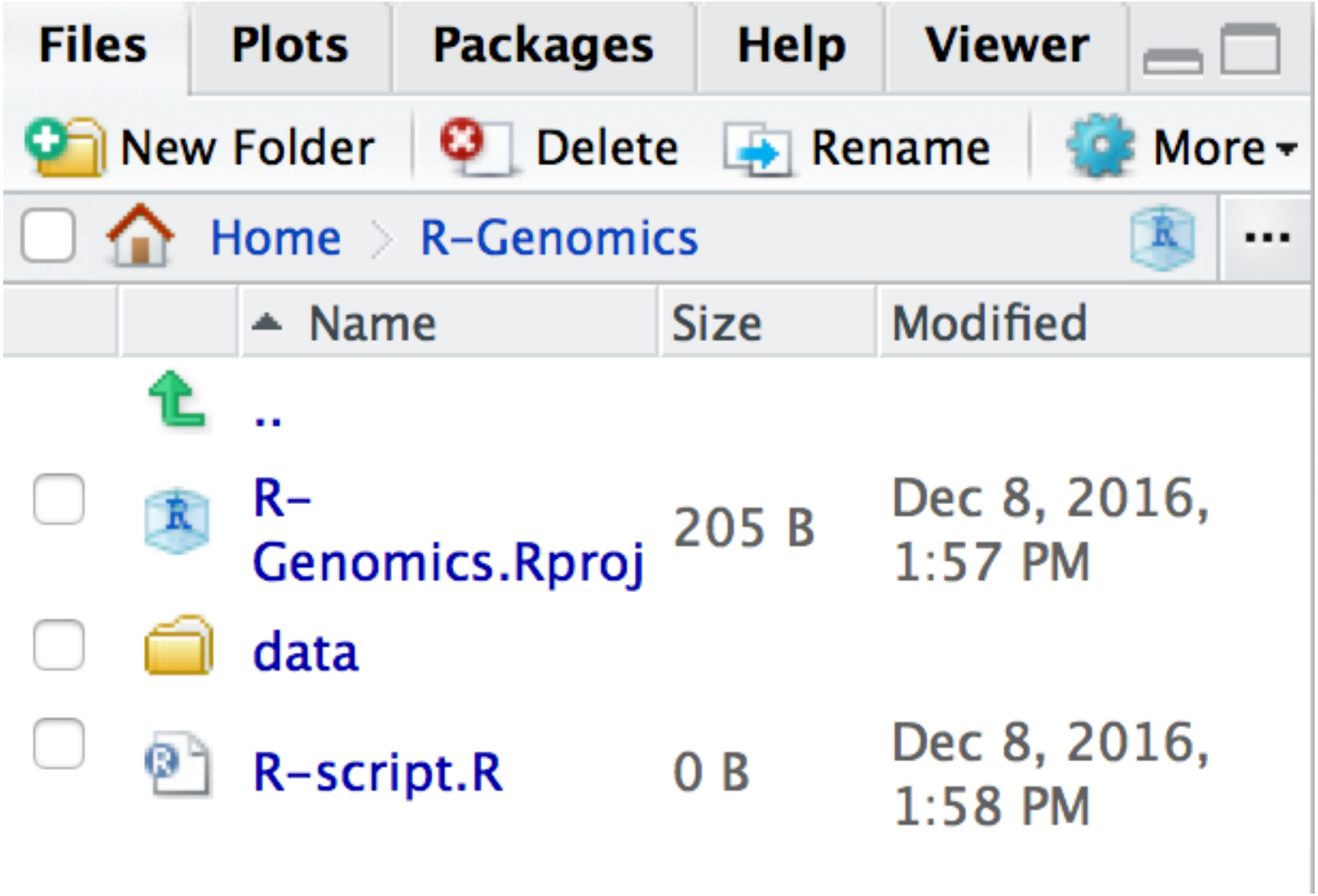
Console

Workspace/History

Info

Before we get started

- Make a new project:
  - File menu -> click on New project -> choose New directory -> then Empty project
- Working directory (`~/R-Genomics`)
- Make a new folder:
  - Files (tab on the right of the screen) -> click on New Folder -> folder name data
- Create a new R script
  - File -> New File -> R-script


Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
R-Genomics
Name
Size
Modified
..
R-Genomics.Rproj
205 B
Dec 8, 2016, 1:57 PM
data
R-script.R
0 B
Dec 8, 2016, 1:58 PM

Basics in R

- Organizing your working directory
  - e.g. raw_data/, figures_output/, data_output/, etc.
- Seeking help:
  - Help with a specific function
    - ?barplot
  - Forgot the arguments?
    - args(lm)
  - Forgot the package?
    - ??geom_point
  - I know to do something but I don't know the function?
    - help.search("kruskal")

Where to ask for help?

- Google it!
- Ask colleagues.
- "R help" mailing list:
  - https://stat.ethz.ch/mailman/listinfo/r-help
- Stackoverflow.
  - Be specific and show that you have already tried hard.
- Tips about how to ask for help:
  - http://blog.revolutionanalytics.com/2014/01/how-to-ask-for-r-help.html

The R syntax

- Use console as a calculator
  - e.g. `3 + 5`
- Use "#" for comments:
  - e.g. `# I am adding 3 and 5. R is fun!`
- Assign the result to a variable by using "<-"
  - e.g. `x <- 3 + 5`

Functions and arguments

- "Canned scripts" that automate something complicated.
- Take inputs as arguments and return values as outputs (not in all cases!).
  - a <- 4

    sqrt(a)

    ## [1] 2
  - round(3.14159)

    ## [1] 3

Lets play a little bit…

- We're going to work with genome lengths
  - Create a variable genome_length_mb and assign it the value 4.6
- Convert this to the weight of the genome in picograms
  - 978Mb = 1picogram
  - Divide the genome length in Mb by 978

Vectors

- A vector is the most common and basic data structure in R.
- A list of values:
  - Numbers
  - Characters
- Many functions to inspect their context:
  - length(a): tells you how many elements are in vector "a"
  - class(a): indicates the type of element of object "a"
  - str(a): provides an overview of the object "a" and the elements it contains

Data types

- Numeric
- Character
- Logical: Boolean (TRUE/FALSE)
- Integer
- Complex: complex numbers with real and imaginary parts (e.g. 1+4i)

Looking at Metadata 1

- Studying a population of Escherichia coli (Ara-3)
  - Propagated for more than 40,000 generations in a glucose-limited minimal medium
  - This medium was supplemented with citrate which E. coli cannot metabolize in the aerobic conditions of the experiment
  - Sequencing of the populations at regular time points reveals that spontaneous citrate-using mutants (Cit+) appeared at around 31,000 generations.
  - This metadata describes information on the Ara-3 clones

Looking at Metadata 2

Column	Description
sample	clone name
generation	generation when sample frozen
clade	based on parsimony-based tree
strain	ancestral strain
cit	citrate-using mutant status
run	Sequence read archive sample ID
genome_size	size in Mbp (made up data for this lesson)

Metadata availability

http://www.datacarpentry.org/R-genomics/data/Ecoli_metadata.csv

Loading Metadata

- Find the working directory
  - getwd()
- Create a new directory named “data”
- Move the downloaded file to “data” folder
- Load the file:
  - `metadata <- read.csv('data/Ecoli_metadata.csv’)`
  - `head(metadata)`

  Or
  - `(metadata <- read.csv('data/Ecoli_metadata.csv’))`

Data.frame

- The *de facto* data structure for most tabular data and what we use for statistics and plotting
- Created by the functions `read.csv()` or `read.table()`

Inspecting data.frame objects

- Size:
  - dim() - returns a vector with the number of rows in the first element, and the number of columns as the second element (the __dim__ensions of the object)
  - nrow() - returns the number of rows
  - ncol() - returns the number of columns
- Content:
  - head() - shows the first 6 rows
  - tail() - shows the last 6 rows
- Names:
  - names() - returns the column names (synonym of colnames() for data.frame objects)
  - rownames() - returns the row names
- Summary:
  - str() - structure of the object and information about the class, length and content of each column
  - summary() - summary statistics for each column

- Note: most of these functions are “generic”, they can be used on other types of objects besides data.frame.

Indexing within a vector

- How to extract one or more values from a vector:
  - metadata[1, 2] *# first element in the 2nd column of the data frame*
  - metadata[1, 6] *# first element in the 6th column*
  - metadata[1:3, 7] *# first three elements in the 7th column*
  - metadata[3,] *# the 3rd element for all columns*
  - metadata[, 7] *# the entire 7th column*
  - head_meta <- metadata[1:6,] *# metadata[1:6,] is equivalent to head(metadata)*

- Remember: R indexes start at 1

“$ sign” for data.frame indexing

- Larger datasets == Difficult to remember the column number
- Use:
  - `names(metadata)`
  - `colnames(metadata)`
- Extract all the info from a column named “strain”
  - `metadata$strain`
- More than one column:
  - `metadata[, c("strain", "clade")]`
- Or a piece of it:
  - `metadata[4:7, c("strain", "clade")]`

Data manipulation using dplyr

- Making data manipulation easier
- Work directly with data frames
- Ability to work with data stored directly in an external database (saves memory)

Install dplyr

- `install.packages("dplyr”)##install`
- `library("dplyr") ## load the library`

Selecting columns and filtering rows

- Powerful functions:
  - select()
  - filter()
  - mutate()
  - group_by()
  - summarize()

Selecting columns and filtering rows

- Choose columns
  - `select(metadata, sample, clade, cit, genome_size)`

- Choose rows:
  - `filter(metadata, cit == "plus")`

Pipes 1

- Combine commands (just like shell….)

```
metadata %>%
  filter(cit == "plus") %>%
  select(sample, generation,
    clade)
```

Pipes 2

```
meta_citplus <- metadata %>%
  filter(cit == "plus") %>%
  select(sample, generation,
         clade)

  meta_citplus
```

Mutate 1

- Put an extra results column

```
metadata %>%
  mutate(genome_bp =
   genome_size *1e6)
```

Mutate 2

- Increase the pipe with head()

```
metadata %>%
    mutate(genome_bp =
      genome_size *1e6)%>%
       head
```

Mutate 3

- Do you have missing data?

```
metadata %>%
  mutate(genome_bp = genome_size
  *1e6) %>% filter(!is.na(clade))
  %>%
  head
```

Save it to a file

- ```
  metadata %>%
  genome_db <- mutate(genome_bp = genome_size *1e6) %>% filter(!is.na(clade))
  ```

- ```
  write.csv(genome_db, file = "data/metadata_gdb.csv")
  ```

Split-apply-combine

- Group your data:

```
metadata %>%
  group_by(cit) %>%
  tally() ##Count
```

Apply statistics functions

```
metadata %>%
  group_by(cit) %>%
   summarize(mean_size =
  mean(genome_size,na.rm =
  TRUE))
```

OR, group by multiple columns

```
metadata %>%
  group_by(cit, clade) %>%
  summarize(mean_size =
  mean(genome_size,na.rm = TRUE))
```

And…filter the missing data

```
metadata %>%
  group_by(cit, clade) %>%
  summarize(mean_size =
  mean(genome_size, na.rm =TRUE))
  %>%
  filter(!is.na(clade))
```

You can also summarize multiple variables at the same time:

```
metadata %>%
  group_by(cit, clade) %>%
  summarize(mean_size =
  mean(genome_size, na.rm =
  TRUE), min_generation =
  min(generation))
```

More cool stuff from dplyr??

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

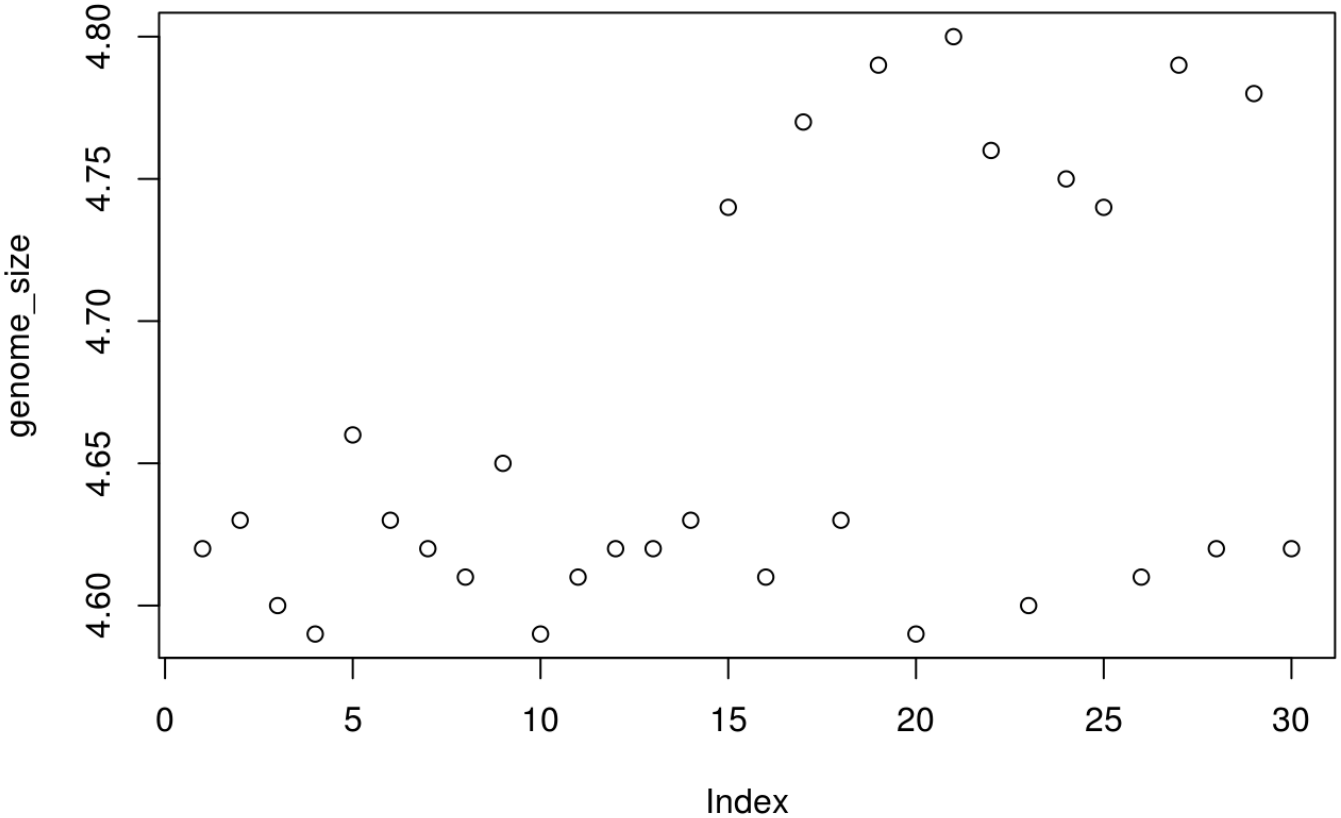
Basic plots in R

"The purpose of computing is insight, not numbers"

Richard Hamming

Scatter plot 1

```
genome_size <- metadata$genome_size
plot(genome_size)
```



Scatter plot 2

- Change the data points:

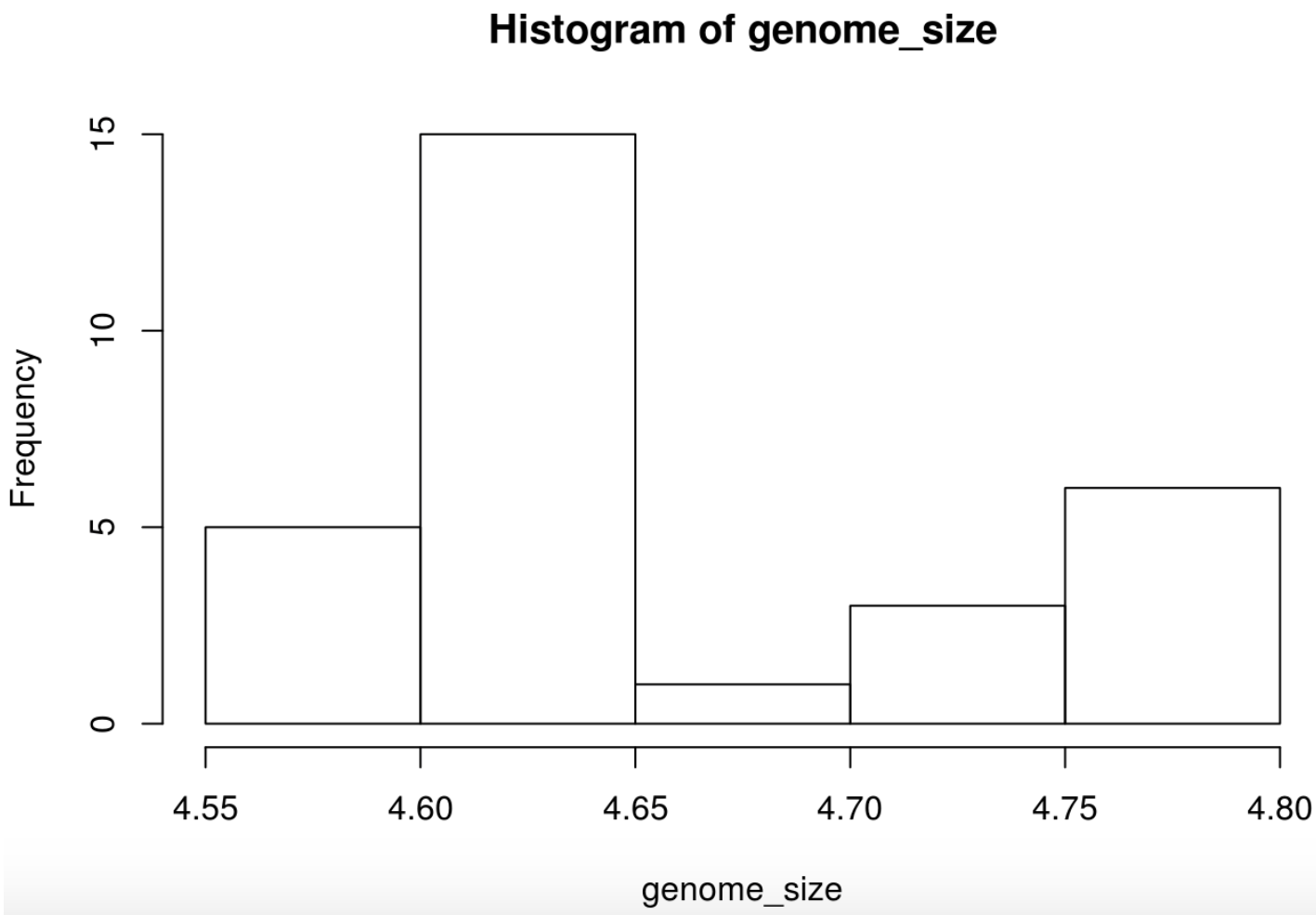
```
plot(genome_size, pch=8)
```

- Put a title to your graph

```
plot(genome_size, pch=8,
main="Scatter plot of genome
sizes")
```

Histogram

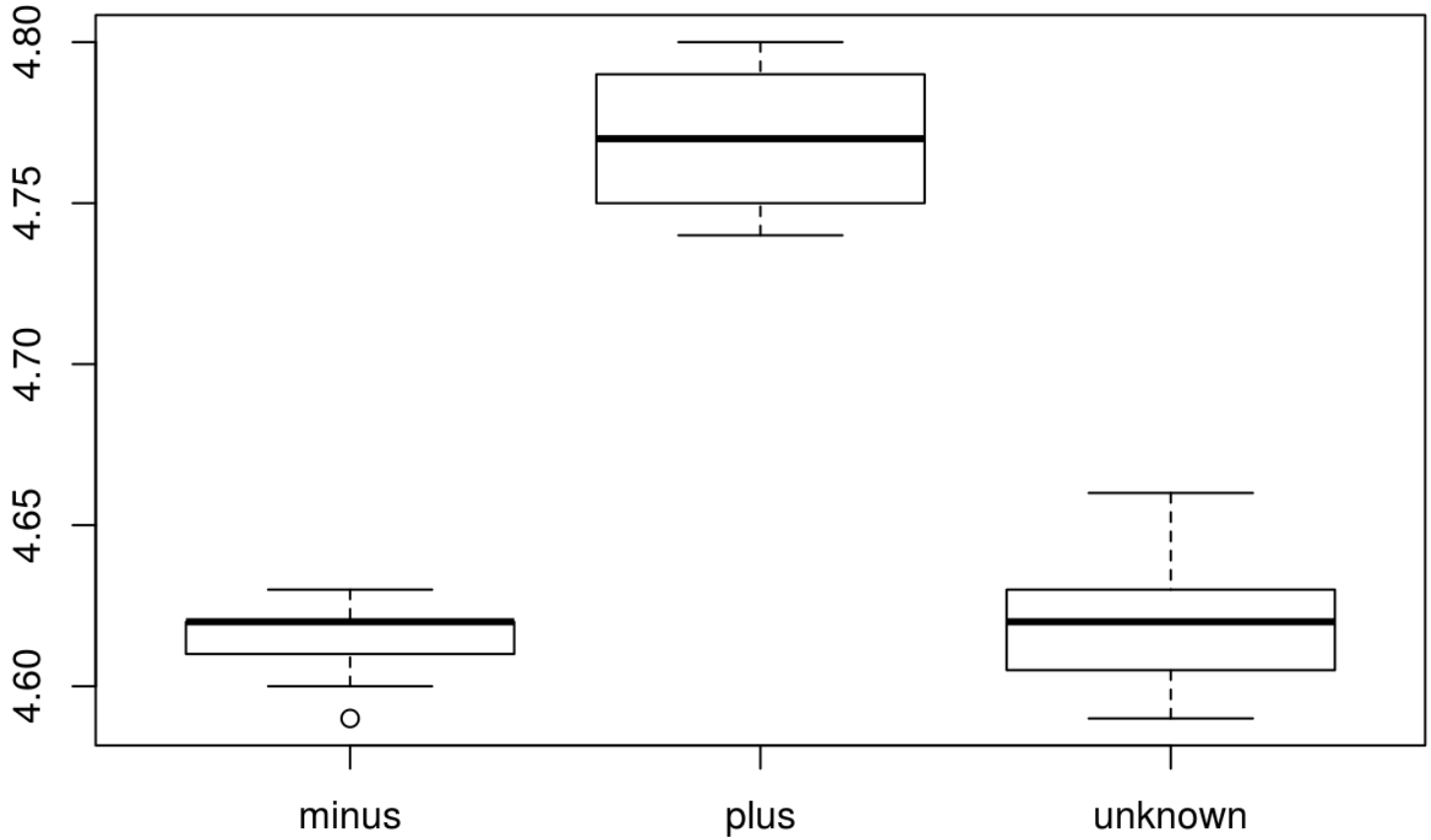
```
hist(genome size)
```



Boxplot 1

- Additional information (cit vector)

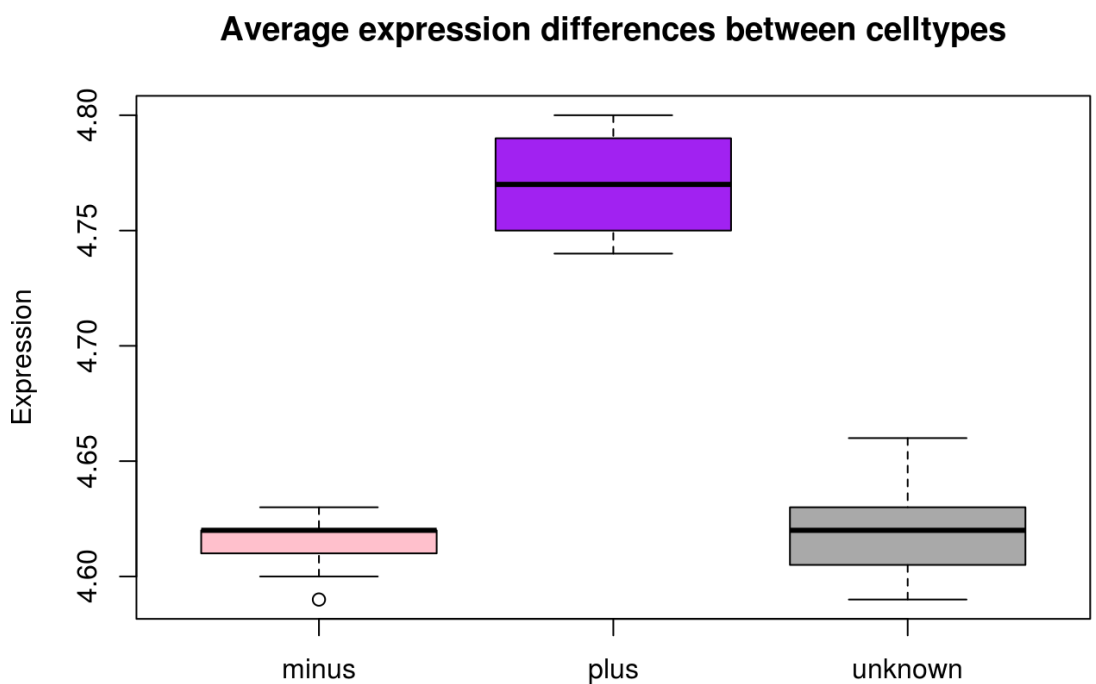
```
boxplot(genome_size ~ cit, metadata)
```



Boxplot 2

- Put some color…

```
boxplot(genome_size ~ cit, metadata,
col=c("pink","purple", "darkgrey"),
main="Average expression differences between
celltypes", ylab="Expression")
```



Export your figures

Console ~/R-Genomics/

```
R version 3.2.4 (2016-03-10) -- "Very Secure Dishes"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (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
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> metadata <- read.csv('data/Ecoli_metadata.csv')
> boxplot(genome_size ~ cit, metadata,  col=c("pink","purple", "darkgrey"), main="Average expre
ssion differences between celltypes", ylab="Expression")
>
>
```

Environment History

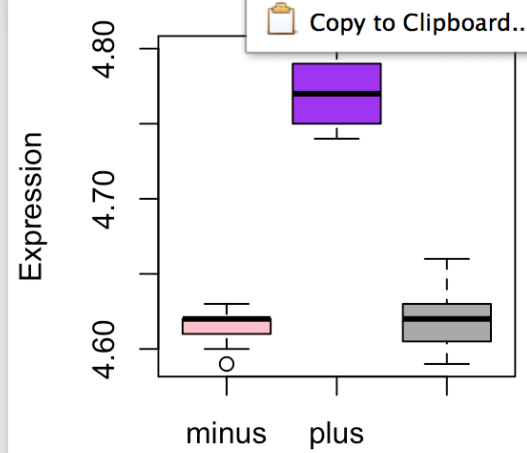
nsole To Source

```
ggplot(metadata) +
geom_point(aes(x = sample, y= genome_size…
theme(axis.text.x = element_text(angle=45…
boxplot <- ggplot(metadata) +
geom_point(aes(x = sample, y= genome_size…
theme(axis.text.x = element_text(angle=45…
pdf("data/boxplot.pdf")
metadata <- read.csv('data/Ecoli_metadata…
boxplot(genome_size ~ cit, metadata, col=…
```

Plots Packages Help Viewer

Zoom Export Publi

Save as Image...
Save as PDF...
Copy to Clipboard...



Choose plot in “info”

Choose the file format that you want

Advanced figures (ggplot2)

- Extremely powerful and flexible plotting package
- Geometric objects (**geom**):
  - points (geom_point, for scatter plots, dot plots, etc)
  - lines (geom_line, for time series, trend lines, etc)
  - boxplot (geom_boxplot, for, well, boxplots!)
- A plot **must have at least one geom**
  - No upper limit
  - Add a **geom** to a plot using the "**+**" operator

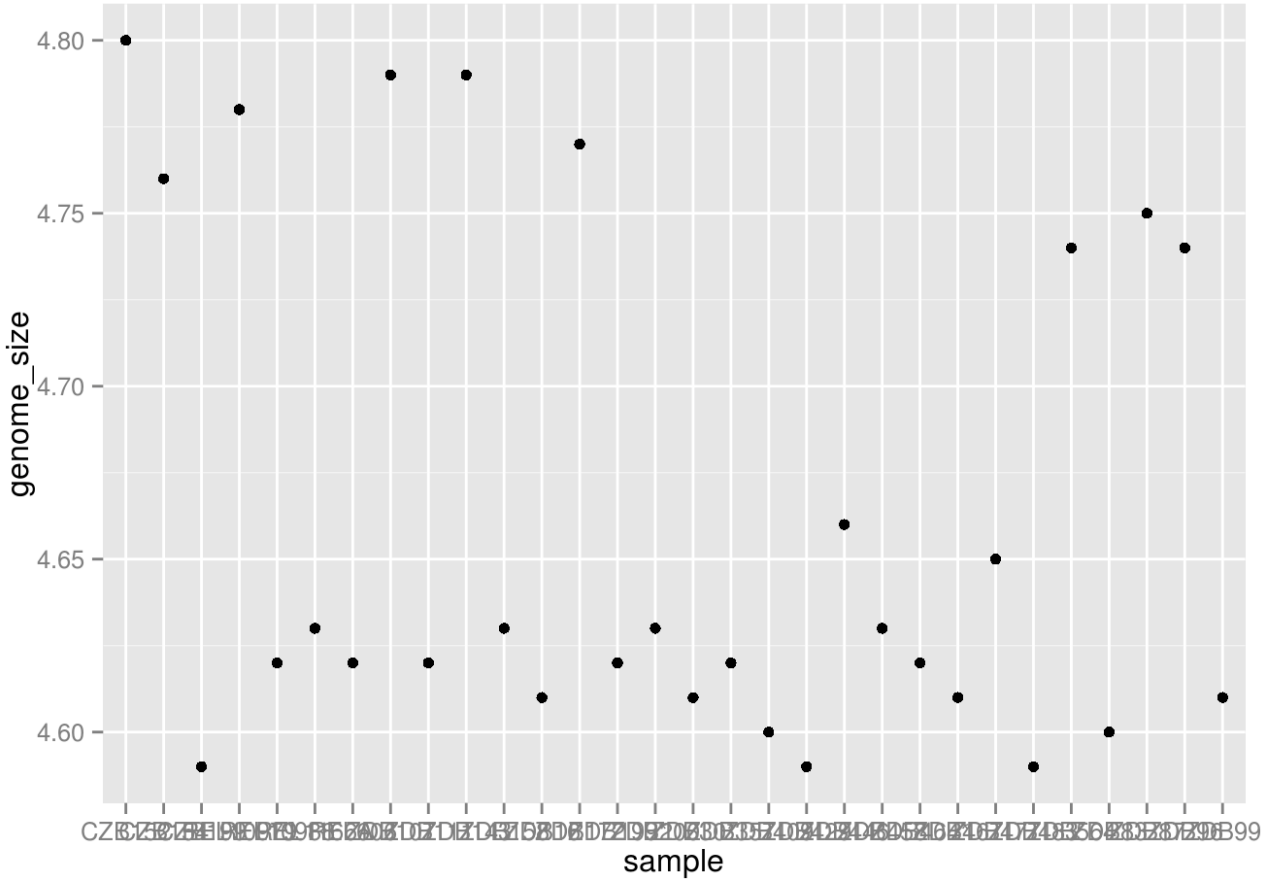
Start with ggplot2

- `library(ggplot2)`
- `ggplot(metadata)` *#note the error*
- `ggplot(metadata) +`

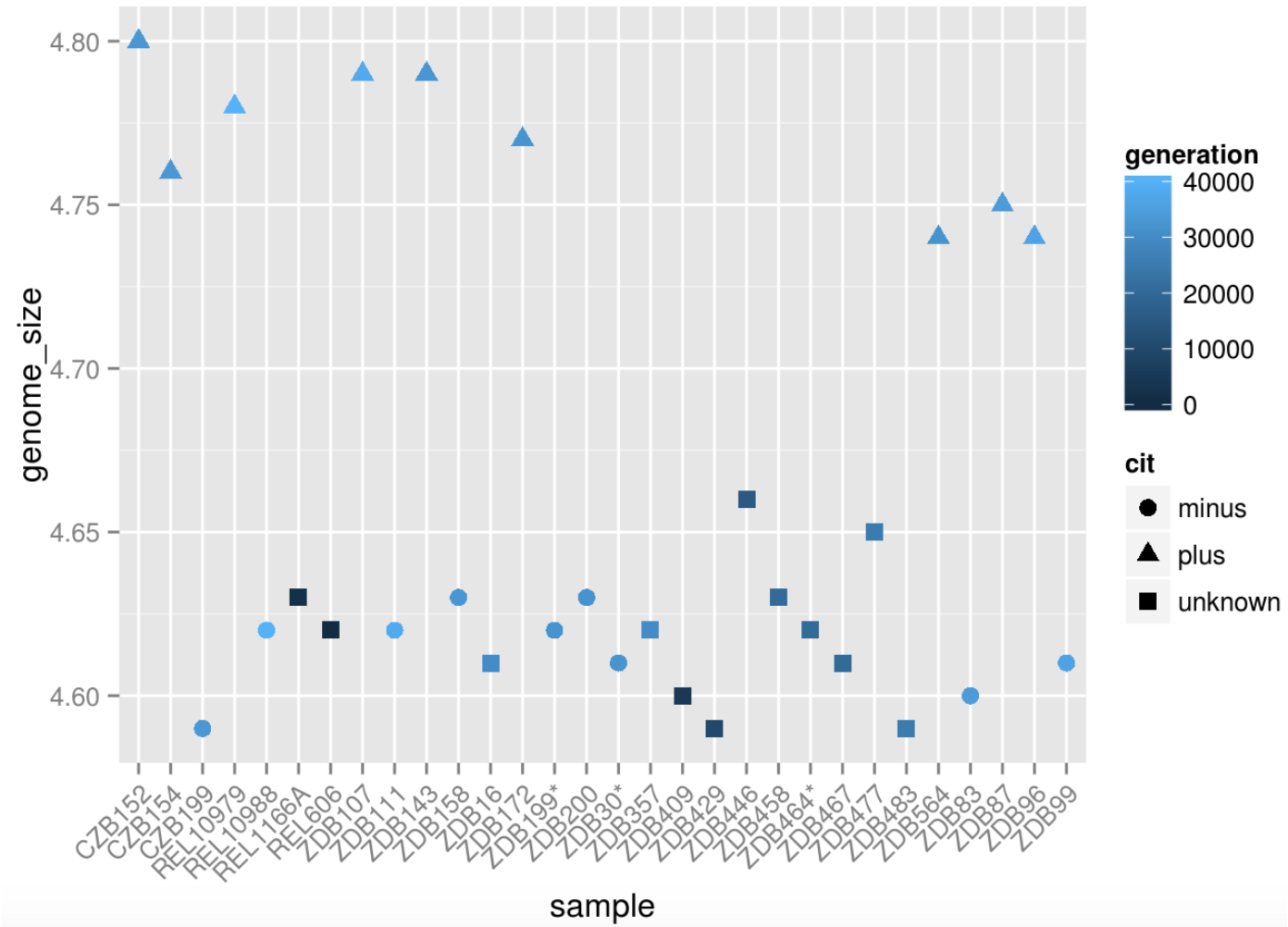
`geom_point()` *#note what happens here*

Simple scatter

```
ggplot(metadata) +
 geom_point(aes(x = sample, y= genome_size))
```



```
ggplot(metadata) +
geom_point(aes(x = sample, y= genome_size, color
= generation, shape = cit), size = rel(3.0)) +
theme(axis.text.x = element_text(angle=45,
hjust=1))
```



Support

- Material on the site:
  - https://bioinformatics.ibers.aber.ac.uk/training/tutorials/#r

- Email:
  - vpl@aber.ac.uk (Vasilis)
  - mis20@aber.ac.uk (Mike)