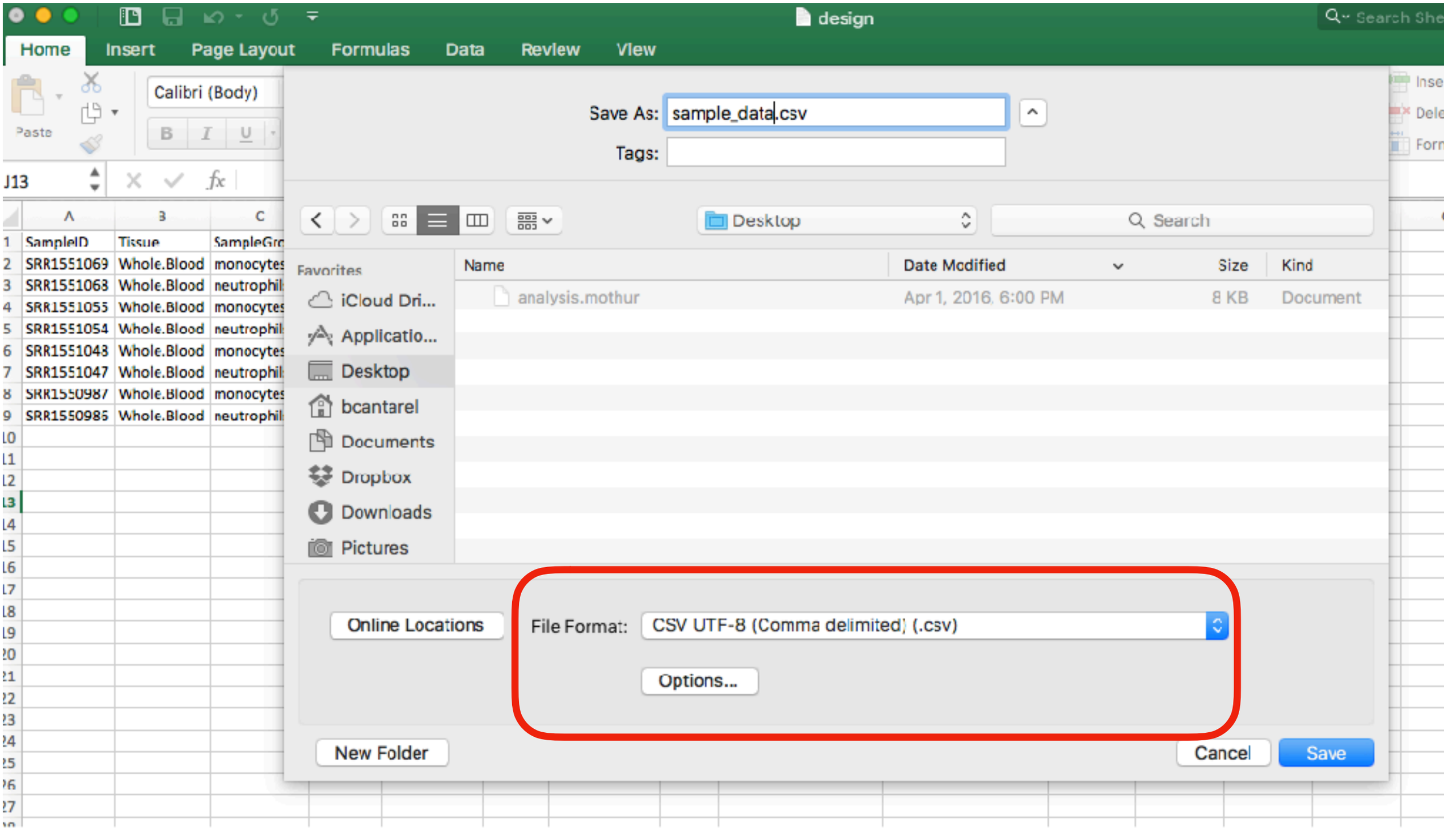
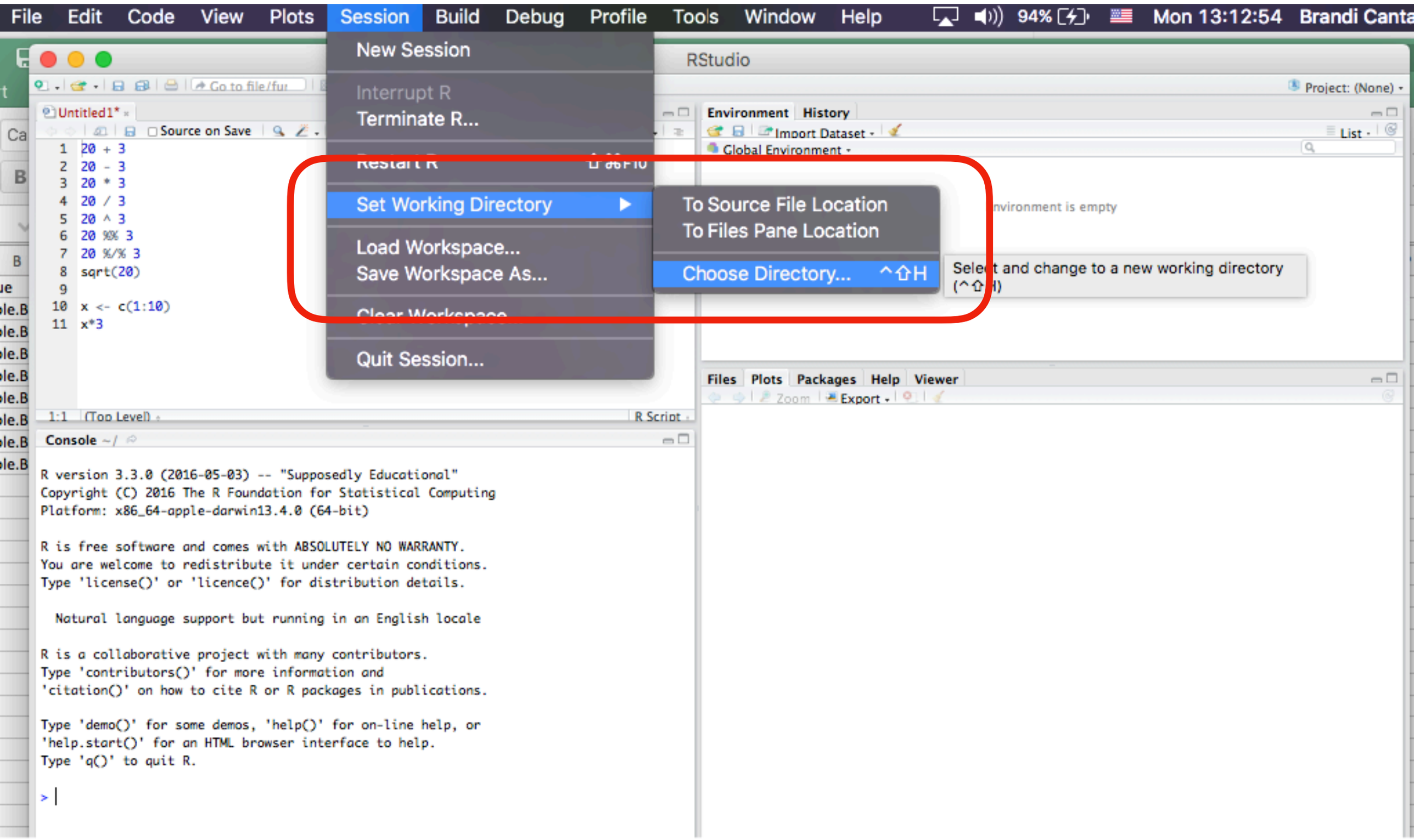
Data Manipulation

R for Beginners 2

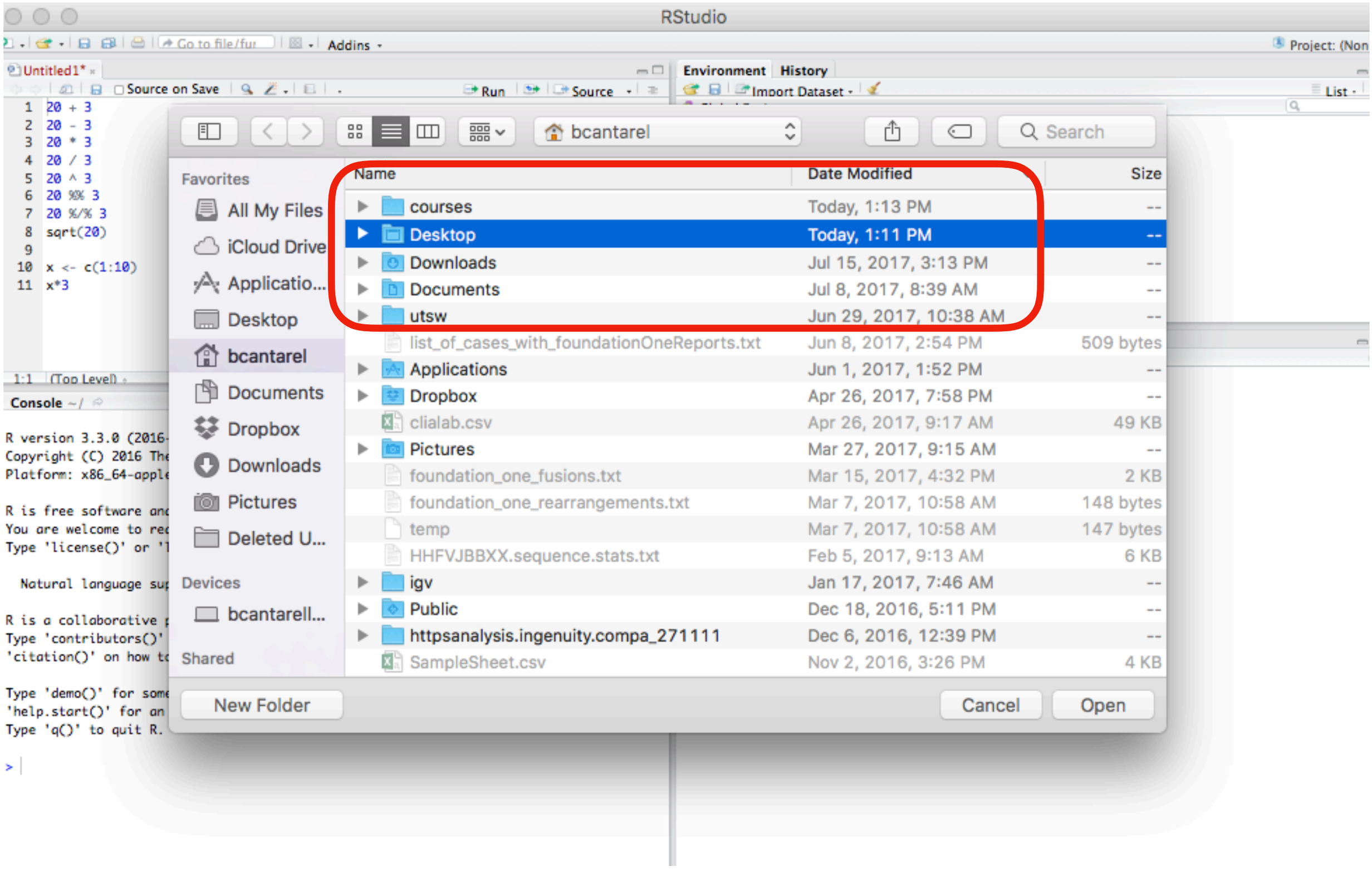
Excel To Data Frame



Excel To Data Frame



Excel To Data Frame



Excel To Data Frame

- `setwd("~/Desktop")`
- `tbl <- read.csv(file="sample_data.csv", header=TRUE)`

```
> head(tbl)
    SampleID      Tissue SampleGroup SubjectID     Organism  Race
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens White
2 SRR1551068 Whole.Blood neutrophils        53 Homo sapiens White
3 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens White
4 SRR1551054 Whole.Blood neutrophils        21 Homo sapiens White
5 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens White
6 SRR1551047 Whole.Blood neutrophils        20 Homo sapiens White
      SampleName Gender        FullPathToFqR1
1   53_Monocytes female SRR1551069_1.fastq.gz
2 53_Neutrophils female SRR1551068_1.fastq.gz
3   21_Monocytes female SRR1551055_1.fastq.gz
4 21_Neutrophils female SRR1551054_1.fastq.gz
5   20_Monocytes female SRR1551048_1.fastq.gz
6 20_Neutrophils female SRR1551047_1.fastq.gz
        FullPathToFqR2
1 SRR1551069_2.fastq.gz
2 SRR1551068_2.fastq.gz
3 SRR1551055_2.fastq.gz
4 SRR1551054_2.fastq.gz
5 SRR1551048_2.fastq.gz
6 SRR1551047_2.fastq.gz
```

Data Frames

- `tbl[3:5]`
  - `columns 3,4,5 of data frame`
- `tbl[c(“SampleID”,”Tissue")]`
  - `columns SampleID and Tissue from data frame`
- `tbl$Gender`
  - `variable Gender in the data frame`
- `tbl[tbl$SampleGroup == ‘monocytes',]`
- `subset(x=tbl,SampleGroup == ‘monocytes',select=c('Tissue','SampleID'))`

R Functions

- Built-in functions are operations that one can “perform” on object that are available in R
- User-defined functions are functions that are written by the user
- Packages are R functions that are written by the R community that need to be loaded before using them

Getting Help with Functions

- R Help: help() and ?
- The help() function and ? help operator in R provide access to the documentation pages for R functions, data sets, and other objects, both for packages in the standard R distribution and for contributed packages.
- To access documentation for the standard lm (linear model)
    - help(lm)
    - help(“lm")
    - ?lm
    - ?"lm" (i.e., the quotes are optional).
- To access help for a function in a package that’s not currently loaded, specify in addition the name of the package: for the rlm() (robust linear model) function in the MASS package:
    - help(rlm, package="MASS")

Data Frame Functions

Operator or Function	Description
str(df)	gives a very brief description of the data
names(df)	gives the name of each variables
summary(df)	gives some very basic summary statistics for each variable
head(df)	shows the first few rows
tail(df)	shows the last few rows.
duplicated()	looks at duplicated elements and returns a logical vector. You can use table() to summarize this vector.
unique()	keeps only the unique lines in a dataset.

Built-In String Functions

Function	Description
substr(*x*, **start=***n1*, **stop=***n2***)**	Extract or replace substrings in a character vector. x <- "abcdef" substr(x, 2, 4) is "bcd" substr(x, 2, 4) <- "22222" is "a222ef"
grep(*pattern*, *x* , **ignore.case=**FALSE, **fixed=**FALSE**)**	Search for *pattern* in *x*. If fixed =FALSE then *pattern* is a regular expression. If fixed=TRUE then *pattern* is a text string. Returns matching indices. grep("A", c("b","A","c"), fixed=TRUE) returns 2
sub(*pattern*, *replacement*, *x*, **ignore.case =**FALSE, **fixed=**FALSE**)**	Find *pattern* in *x* and replace with *replacement* text. If fixed=FALSE then *pattern* is a regular expression. If fixed = T then *pattern* is a text string. sub("\\s",".","Hello There") returns "Hello.There"
strsplit(*x*, *split***)**	Split the elements of character vector *x* at *split*. strsplit("abc", "") returns 3 element vector "a","b","c"
paste(..., sep="")	Concatenate strings after using *sep* string to seperate them. paste("x",1:3,sep="") returns c("x1","x2" "x3") paste("x",1:3,sep="M") returns c("xM1","xM2" "xM3") paste("Today is", date())
toupper(*x***)**	Uppercase
tolower(*x***)**	Lowercase

Built-In Functions

Function	Description
seq(*from* **,** *to***,** *by***)**	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)
rep(*x***,** *ntimes***)**	repeat *x n* times y <- rep(1:3, 2) # y is c(1, 2, 3, 1, 2, 3)
cut(*x***,** *n***)**	divide continuous variable in factor with *n* levels y <- cut(x, 5)

Manipulation of Strings

- `rep(3,'a')`
  - `[1] "a" "a" "a"`
- `a <- "Hello"`
- `b <- 'How'`
- `c <- “are you? "`
- `print(paste(a,b,c))`
  - `[1] "Hello How are you? "`
- `print(paste(a,b,c, sep = "-"))`
  - `[1] "Hello-How-are you? "`
- `print(paste(a,b,c, sep = "", collapse = ""))`
  - `[1] "HelloHoware you? “`
- `toupper(a)`
  - `[1] "HELLO"`
- `tolower(a)`
  - `[1] “hello"`
- `substring(a,1,2)`
  - `[1] "He"`
- `substring(a,2,5)`
  - `[1] "ello"`

Merge Data Frames

```
setwd(“~/Desktop")
tbl1 <- read.csv(file="sample_data.csv",header=TRUE)
tbl2 <- read.csv(file="table2.csv",header=TRUE)
```

```
> head(tbl1)
    SampleID      Tissue SampleGroup SubjectID     Organism  Race     SampleName
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens White   53_Monocytes
2 SRR1551068 Whole.Blood neutrophils        53 Homo sapiens White 53_Neutrophils
3 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens White   21_Monocytes
4 SRR1551054 Whole.Blood neutrophils        21 Homo sapiens White 21_Neutrophils
5 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens White   20_Monocytes
6 SRR1551047 Whole.Blood neutrophils        20 Homo sapiens White 20_Neutrophils
  Gender        FullPathToFqR1        FullPathToFqR2
1 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz
2 female SRR1551068_1.fastq.gz SRR1551068_2.fastq.gz
3 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz
4 female SRR1551054_1.fastq.gz SRR1551054_2.fastq.gz
5 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz
6 female SRR1551047_1.fastq.gz SRR1551047_2.fastq.gz
> head(tbl2)
    SampleID SubjectID BMI
1 SRR1551069        53  23
2 SRR1551068        53  23
3 SRR1551055        21  28
4 SRR1551054        21  28
5 SRR1551048        20  35
6 SRR1551047        20  35
```

Merge Data Frames

```
setwd(“~/Desktop")
tbl1 <- read.csv(file="sample_data.csv",header=TRUE)
tbl2 <- read.csv(file=“table2.csv",header=TRUE)
merge(tbl1,tbl2,by='SampleID')
```

```
> merge(tbl1,tbl2,by='SampleID')
    SampleID      Tissue SampleGroup SubjectID.x     Organism     Race     SampleName
1 SRR1550986 Whole.Blood neutrophils          44 Homo sapiens Hispanic 44_Neutrophils
2 SRR1550987 Whole.Blood   monocytes          44 Homo sapiens Hispanic   44_Monocytes
3 SRR1551047 Whole.Blood neutrophils          20 Homo sapiens    White 20_Neutrophils
4 SRR1551048 Whole.Blood   monocytes          20 Homo sapiens    White   20_Monocytes
5 SRR1551054 Whole.Blood neutrophils          21 Homo sapiens    White 21_Neutrophils
6 SRR1551055 Whole.Blood   monocytes          21 Homo sapiens    White   21_Monocytes
7 SRR1551068 Whole.Blood neutrophils          53 Homo sapiens    White 53_Neutrophils
8 SRR1551069 Whole.Blood   monocytes          53 Homo sapiens    White   53_Monocytes
  Gender        FullPathToFqR1        FullPathToFqR2 SubjectID.y BMI
1 female SRR1550986_1.fastq.gz SRR1550986_2.fastq.gz          44  31
2 female SRR1550987_1.fastq.gz SRR1550987_2.fastq.gz          44  31
3 female SRR1551047_1.fastq.gz SRR1551047_2.fastq.gz          20  35
4 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz          20  35
5 female SRR1551054_1.fastq.gz SRR1551054_2.fastq.gz          21  28
6 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz          21  28
7 female SRR1551068_1.fastq.gz SRR1551068_2.fastq.gz          53  23
8 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz          53  23
```

Merge Data Frames

Description

Merge two data frames by common columns or row names, or do other versions of database *join* operations.

Usage

```
merge(x, y, ...)

## Default S3 method:
merge(x, y, ...)

## S3 method for class 'data.frame'
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"),
      incomparables = NULL, ...)
```

Arguments

`x, y`	data frames, or objects to be coerced to one.
`by, by.x, by.y`	specifications of the columns used for merging. See 'Details'.
`all`	logical; `all = L` is shorthand for `all.x = L` and `all.y = L`, where `L` is either `TRUE` or `FALSE`.
`all.x`	logical; if `TRUE`, then extra rows will be added to the output, one for each row in `x` that has no matching row in `y`. These rows will have `NA`s in those columns that are usually filled with values from `y`. The default is `FALSE`, so that only rows with data from both `x` and `y` are included in the output.
`all.y`	logical; analogous to `all.x`.
`sort`	logical. Should the result be sorted on the `by` columns?
`suffixes`	a character vector of length 2 specifying the suffixes to be used for making unique the names of columns in the result which not used for merging (appearing in `by` etc).
`incomparables`	values which cannot be matched. See `match`. This is intended to be used for merging on one column, so these are incomparable values of that column.
`...`	arguments to be passed to or from methods.

Merge Data Frames

Examples

```
## use character columns of names to get sensible sort order
authors <- data.frame(
    surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
    nationality = c("US", "Australia", "US", "UK", "Australia"),
    deceased = c("yes", rep("no", 4)))
books <- data.frame(
    name = I(c("Tukey", "Venables", "Tierney",
               "Ripley", "Ripley", "McNeil", "R Core")),
    title = c("Exploratory Data Analysis",
              "Modern Applied Statistics ...",
              "LISP-STAT",
              "Spatial Statistics", "Stochastic Simulation",
              "Interactive Data Analysis",
              "An Introduction to R"),
    other.author = c(NA, "Ripley", NA, NA, NA, NA,
                     "Venables & Smith"))

(m1 <- merge(authors, books, by.x = "surname", by.y = "name"))
(m2 <- merge(books, authors, by.x = "name", by.y = "surname"))
stopifnot(as.character(m1[, 1]) == as.character(m2[, 1]),
          all.equal(m1[, -1], m2[, -1][ names(m1)[-1] ]),
          dim(merge(m1, m2, by = integer(0))) == c(36, 10))

## "R core" is missing from authors and appears only here :
merge(authors, books, by.x = "surname", by.y = "name", all = TRUE)

## example of using 'incomparables'
x <- data.frame(k1 = c(NA,NA,3,4,5), k2 = c(1,NA,NA,4,5), data = 1:5)
y <- data.frame(k1 = c(NA,2,NA,4,5), k2 = c(NA,NA,3,4,5), data = 1:5)
merge(x, y, by = c("k1","k2")) # NA's match
merge(x, y, by = "k1") # NA's match, so 6 rows
merge(x, y, by = "k2", incomparables = NA) # 2 rows
```

sqldf : R package for running SQL statements on R data frames

- `head(tbl)`
  - `read.csv.sql(tbl1.filename, sql = sqldf("select * from file limit 6”),sep=',')`
- `tbl$Gender`
  - `read.csv.sql(tbl1.filename, sql = paste("select Gender from file"),sep=',')`
- `tbl[c(“SampleID”,”Tissue")]`
  - `sqldf("select SampleID,Tissue from file”)`
- `tbl[tbl$SampleGroup == ‘monocytes’,]`
  - `sqldf("select * from file where SampleGroup = ‘monocyte’”)`
- `subset(x=tbl,SampleGroup == ‘monocytes’,select=c('Tissue','SampleID'))`
  - `read.csv.sql(tbl1.filename, sql = paste("select Tissue,SampleID from file where SampleGroup = 'monocytes'",sep=''),sep = ",")`

Relational Databases

- Using relational databases you can:
    - join multiple tables
    - filter and sort
    - perform mathematical calculations
    - find fuzzy matches

Create Some Tables

```
CREATE TABLE subject (
  subjid int(10) NOT NULL auto_increment,
  subjacc varchar(100),
  tissue varchar(100),
  tumortype varchar(100),
  index(subjid),
  index(subjacc),
  index(tumortype),
  index(tissue)
) ENGINE=MyISAM;
```

```
CREATE TABLE samples (
  sid int(10) NOT NULL auto_increment,
  sample_id varchar(100),
  sample_name varchar(100),
  subjid int(10),
  dnaextractid int(10),
  rnaextractid int(10),
  nuctype varchar(100),
  assay varchar(10),
  index(sid),
  index(sample_id),
  index(sample_name),
  index(subjid),
  index(assay)
) ENGINE=MyISAM;
```

Normalization

The goal of database normalization is to decompose relations with anomalies in order to produce smaller, well-structured relations. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

Title	Author	Bio	ISBN	Subject	Pages	Publisher
Beginning MySQL Database Design and Optimization	Chad Russell, Jon Stephens	Chad Russell is a programmer and network administrator who owns his own Internet hosting company., Jon Stephens is a member of the MySQL AB documentation team.	1590593324	MySQL, Database Design	520	Apress

ISBN	Title	Pages
1590593324	Beginning MySQL Database Design and Optimization	520

Author_ID	First_Name	Last_name
1	Chad	Russell
2	Jon	Stephens
3	Mike	Hillyer

Subject_ID	Name
1	MySQL
2	Database Design

Publisher_ID	Name	Address	City	State	Zip
1	Apress	2560 Ninth Street, Station 219	Berkeley	California	94710

auto_increment adds a unique id to every row

```
CREATE TABLE animals (
     id MEDIUMINT NOT NULL AUTO_INCREMENT,
     name CHAR(30) NOT NULL,
     PRIMARY KEY (id)
) ENGINE=MyISAM;

INSERT INTO animals (name) VALUES
    ('dog'),('cat'),('penguin'),
    ('lax'),('whale'),('ostrich');

SELECT * FROM animals;
```

Which returns:

```
+----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | lax     |
|  5 | whale   |
|  6 | ostrich |
+----+---------+
```

Pick a database

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| SequencingCenter   |
| genomeseer         |
| mysql              |
| performance_schema |
| validation_db      |
+--------------------+
6 rows in set (0.00 sec)

MariaDB [(none)]> use validation_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

Describe and Show Tables

```
mysql > show tables;
+--------------------------+
| Tables_in_validation_db |
+--------------------------+
| coverage                 |
| demultiplex              |
| dna_alignment            |
| dna_extract              |
| dna_hyb                  |
| dna_lib_prep             |
| dna_lib_prep_dna_hyb     |
| exon                     |
| nuc_extract              |
| rna_alignment            |
| rna_extract              |
| rna_lib_prep             |
| sample                   |
| seq_run                  |
| subject                  |
+--------------------------+
15 rows in set (0.00 sec)
```

```
MariaDB [validation_db]> describe subject;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| subject_id  | int(11)     | NO   | PRI | NULL    |       |
| subject_acc | varchar(255)| YES  |     | NULL    |       |
| tissue      | varchar(255)| YES  |     | NULL    |       |
| tumor_type  | varchar(255)| YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Load Data, Insert Row

```
Whistler        Gwen    bird    \N      1997-12-09      \N
```

To load the text file `pet.txt` into the **`pet`** table, use this statement:

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet;
```

If you created the file on Windows with an editor that uses **`\r\n`** as a line terminator, you should use this statement instead:

```
mysql> LOAD DATA LOCAL INFILE '/path/pet.txt' INTO TABLE pet
    -> LINES TERMINATED BY '\r\n';
```

(On an Apple machine running OS X, you would likely want to use **`LINES TERMINATED BY '\r'`**.)

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number LINES]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Queries

```
mysql > select * from subject limit 10;
+------------+-------------+------------+----------------------+
| subject_id | subject_acc | tissue     | tumor_type           |
+------------+-------------+------------+----------------------+
|          1 | 12546       | Kidney     | ClearCellKidneyCancer |
|          2 | 24007       | Kidney     | ClearCellKidneyCancer |
|          3 | 24329       | Kidney     | ClearCellKidneyCancer |
|          4 | 25516       | Kidney     | ClearCellKidneyCancer |
|          5 | 25838       | Kidney     | ClearCellKidneyCancer |
|          6 | 26473       | Kidney     | ClearCellKidneyCancer |
|          7 | 27522       | Kidney     | ClearCellKidneyCancer |
|          8 | AP13-1125   | Breast     | Breast ILC           |
|         10 | AP14-924    | Breast     | Breast ILC           |
|         11 | APL         | BoneMarrow | APL                  |
+------------+-------------+------------+----------------------+
10 rows in set (0.00 sec)
```

```
> select subject_id,tissue from subject limit 10;
+------------+------------+
| subject_id | tissue     |
+------------+------------+
|          1 | Kidney     |
|          2 | Kidney     |
|          3 | Kidney     |
|          4 | Kidney     |
|          5 | Kidney     |
|          6 | Kidney     |
|          7 | Kidney     |
|          8 | Breast     |
|         10 | Breast     |
|         11 | BoneMarrow |
+------------+------------+
10 rows in set (0.00 sec)
```

```
> select * from subject where tissue = 'Kidney' limit 10;
+------------+-------------+--------+----------------------+
| subject_id | subject_acc | tissue | tumor_type           |
+------------+-------------+--------+----------------------+
|          1 | 12546       | Kidney | ClearCellKidneyCancer |
|          2 | 24007       | Kidney | ClearCellKidneyCancer |
|          3 | 24329       | Kidney | ClearCellKidneyCancer |
|          4 | 25516       | Kidney | ClearCellKidneyCancer |
|          5 | 25838       | Kidney | ClearCellKidneyCancer |
|          6 | 26473       | Kidney | ClearCellKidneyCancer |
|          7 | 27522       | Kidney | ClearCellKidneyCancer |
|         73 | SU15-1385   | Kidney | TCCA Kidney          |
|        133 | Z12-2637    | Kidney | Renal Cell Carcinoma |
|        134 | Z13-1093    | Kidney | Renal Cell Carcinoma |
+------------+-------------+--------+----------------------+
```

```
> select * from dna_extract where extract_din < 2.5 limit 5;
+---------------+---------------+-------------+--------------+------+
| dna_extract_id | nuc_extract_id | extract_din | dna_yield_ng | pass |
+---------------+---------------+-------------+--------------+------+
|             10 |            66 |         2.4 |           89 |    0 |
|             13 |           194 |         2.3 |           92 |    0 |
|             21 |           170 |           0 |            7 |    0 |
|             22 |           165 |           0 |           12 |    0 |
|             23 |           166 |           0 |           12 |    0 |
+---------------+---------------+-------------+--------------+------+
5 rows in set (0.00 sec)
```

MySQL can tell time, do math, etc

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+------------------+---------+
| SIN(PI()/4)      | (4+1)*5 |
+------------------+---------+
| 0.70710678118655 |      25 |
+------------------+---------+
1 row in set (0.02 sec)
```

The queries shown thus far have been relatively short, single-line statements. You can even enter multiple statements on a single line. Just end each one with a semicolon:

```
mysql> SELECT VERSION(); SELECT NOW();
+----------------+
| VERSION()      |
+----------------+
| 5.0.7-beta-Max |
+----------------+
1 row in set (0.00 sec)

+---------------------+
| NOW()               |
+---------------------+
| 2005-07-11 17:59:36 |
+---------------------+
1 row in set (0.00 sec)
```

Counting, sorting and grouping

```
> select count(*) from subject
where tissue = 'Kidney' limit 10;
+----------+
| count(*) |
+----------+
|       11 |
+----------+
1 row in set (0.00 sec)
```

```
> select tissue,count(*) from subject group by tissue limit 10;
+----------------+----------+
| tissue         | count(*) |
+----------------+----------+
| NULL           |       13 |
| Abdominal wall |        1 |
| AdrenalGland   |        3 |
| BileDuct       |        1 |
| bladder        |        3 |
| Blood          |       26 |
| BoneMarrow     |       45 |
| Brain          |        1 |
| Breast         |        8 |
| CellLine       |       10 |
+----------------+----------+
10 rows in set (0.00 sec)
```

```
> select * from dna_extract where extract_din < 2.5 order by extract_din limit 5;
+---------------+---------------+-------------+--------------+------+
| dna_extract_id | nuc_extract_id | extract_din | dna_yield_ng | pass |
+---------------+---------------+-------------+--------------+------+
|            235 |            251 |           0 |          192 |    0 |
|            170 |             28 |           0 |           27 |    0 |
|            163 |            156 |           0 |           43 |    0 |
|            150 |             40 |           0 |           50 |    0 |
|            192 |             46 |           0 |           83 |    0 |
+---------------+---------------+-------------+--------------+------+
5 rows in set (0.00 sec)
```

Calculations

```
mysql> SELECT name, birth, CURDATE(),
    -> (YEAR(CURDATE())-YEAR(birth))
    -> - (RIGHT(CURDATE(),5)<RIGHT(birth,5))
    -> AS age
    -> FROM pet;
+----------+------------+------------+------+
| name     | birth      | CURDATE()  | age  |
+----------+------------+------------+------+
| Fluffy   | 1993-02-04 | 2003-08-19 |   10 |
| Claws    | 1994-03-17 | 2003-08-19 |    9 |
| Buffy    | 1989-05-13 | 2003-08-19 |   14 |
| Fang     | 1990-08-27 | 2003-08-19 |   12 |
| Bowser   | 1989-08-31 | 2003-08-19 |   13 |
| Chirpy   | 1998-09-11 | 2003-08-19 |    4 |
| Whistler | 1997-12-09 | 2003-08-19 |    5 |
| Slim     | 1996-04-29 | 2003-08-19 |    7 |
| Puffball | 1999-03-30 | 2003-08-19 |    4 |
+----------+------------+------------+------+
```

```
mysql> select patient_id, sex, patient.age, height, weight, hip, bmi
    -> from patient right join sample using(patient_id)
    -> where patient.age = (select max(age) from patient);
+------------+------+------+--------+--------+------+------+
| patient_id | sex  | age  | height | weight | hip  | bmi  |
+------------+------+------+--------+--------+------+------+
|        100 |    2 |   79 |  151.7 |   77.3 |  119 | 33.6 |
+------------+------+------+--------+--------+------+------+
1 row in set (0.00 sec)
```

```
mysql> select bmi_group, max(age) from patient group by bmi_group;
+-----------+----------+
| bmi_group | max(age) |
+-----------+----------+
| L         |       77 |
| OB        |       70 |
| OM        |       79 |
| OvB       |       76 |
| OvM       |       74 |
+-----------+----------+
5 rows in set (0.00 sec)
```

And
Or

```
mysql> select * from patient where age < 40 and bmi_group = 'L' limit 10;
+------------+----------+----------+----------+----------+-----------------+-----+-----------+-----+-------------+-------+---------+---------+--------+---------+-------------+
| patient_id | index_um | agdb4_id | papi_acc | hmp_acc  | num_collections | sex | bmi_group | age | first_study | ms_bp | ms_gluc | ms_hdl | ms_tg | ms_wst | delta_visit |
+------------+----------+----------+----------+----------+-----------------+-----+-----------+-----+-------------+-------+---------+--------+-------+--------+-------------+
|        138 | A02875   |    27833 | P0451    | HMP0213  |               1 |   2 | L         |  26 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|        150 | A02961   |    26494 | P0552    |          |               1 |   1 | L         |  31 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|        161 | A03041   |    50930 | P0046    | HMP0086  |               2 |   1 | L         |  30 | HMP         |     0 |       0 |      0 |     0 |      0 |           4 |
|        162 | A03042   |    41448 | P0047    | HMP0087  |               2 |   2 | L         |  28 | HMP         |     0 |       0 |      0 |     0 |      0 |           4 |
|        169 | A03142   |    54223 | P0544    | HMP0006  |               2 |   1 | L         |  36 | PAPI        |     0 |       0 |      0 |     0 |      0 |           9 |
|        181 | A03349   |    56745 | P0380    | HMP0068  |               2 |   2 | L         |  38 | PAPI        |     0 |       0 |      0 |     0 |      0 |          22 |
|        184 | A03366   |    47959 | P0389    | HMP0231  |               1 |   2 | L         |  28 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|        193 | A03451   |    56226 | P0612    | HMP0269  |               1 |   1 | L         |  22 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|        197 | A03462   |    17429 | P0499    | HMP0018  |               2 |   1 | L         |  39 | PAPI        |     0 |       0 |      0 |     0 |      0 |          13 |
|        198 | A03463   |    53846 | P0498    | HMP0222  |               1 |   2 | L         |  39 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
+------------+----------+----------+----------+----------+-----------------+-----+-----------+-----+-------------+-------+---------+--------+-------+--------+-------------+
10 rows in set (0.00 sec)



mysql> select * from patient where age < 40 or bmi_group =  L' limit 10;
+------------+----------+----------+----------+----------+-----------------+-----+-----------+-----+-------------+-------+---------+--------+-------+--------+-------------+
| patient_id | index_um | agdb4_id | papi_acc | hmp_acc  | num_collections | sex | bmi_group | age | first_study | ms_bp | ms_gluc | ms_hdl | ms_tg | ms_wst | delta_visit |
+------------+----------+----------+----------+----------+-----------------+-----+-----------+-----+-------------+-------+---------+--------+-------+--------+-------------+
|        105 | A02070   |    33700 | P0477    |          |               1 |   2 | L         |  70 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|         11 | A00260   |    15196 |          | HMP0089  |               1 |   1 | L         |  42 | HMP         |     0 |       0 |      0 |     0 |      0 |        NULL |
|        113 | A02241   |    50182 | P0500    | HMP0046  |               2 |   2 | L         |  44 | PAPI        |     0 |       0 |      0 |     0 |      0 |          14 |
|        115 | A02298   |    53328 | P0581    | HMP0032  |               2 |   2 | L         |  58 | PAPI        |     1 |       1 |      0 |     0 |      0 |           8 |
|        119 | A02313   |    53831 |          | HMP0014  |               2 |   2 | L         |  73 | HMP         |     0 |       0 |      0 |     0 |      0 |           4 |
|        127 | A02434   |    62122 | P0630    |          |               1 |   2 | L         |  61 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|        133 | A02673   |    10919 | P0442    | HMP0062  |               2 |   1 | L         |  62 | PAPI        |     0 |       0 |      0 |     0 |      0 |          18 |
|        135 | A02829   |    49252 | P0417    |          |               1 |   2 | OM        |  31 | PAPI        |     1 |       0 |      0 |     0 |      1 |        NULL |
|        138 | A02875   |    27833 | P0451    | HMP0213  |               1 |   2 | L         |  26 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
|        141 | A02895   |    18131 | P0490    | HMP0254  |               1 |   1 | L         |  64 | PAPI        |     0 |       0 |      0 |     0 |      0 |        NULL |
+------------+----------+----------+----------+----------+-----------------+-----+-----------+-----+-------------+-------+---------+--------+-------+--------+-------------+
10 rows in set (0.00 sec)
```

Like

```
mysql> SELECT * FROM pet WHERE name LIKE 'b%';
+--------+--------+---------+------+------------+------------+
| name   | owner  | species | sex  | birth      | death      |
+--------+--------+---------+------+------------+------------+
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | dog     | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+---------+------+------------+------------+
```

To find names ending with "**fy**":

```
mysql> SELECT * FROM pet WHERE name LIKE '%fy';
+--------+--------+---------+------+------------+-------+
| name   | owner  | species | sex  | birth      | death |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | cat     | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | dog     | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

To find names containing a "**w**":

```
mysql> SELECT * FROM pet WHERE name LIKE '%w%';
+----------+-------+---------+------+------------+------------+
| name     | owner | species | sex  | birth      | death      |
+----------+-------+---------+------+------------+------------+
| Claws    | Gwen  | cat     | m    | 1994-03-17 | NULL       |
| Bowser   | Diane | dog     | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen  | bird    | NULL | 1997-12-09 | NULL       |
+----------+-------+---------+------+------------+------------+
```

Joining

JOIN: Return rows when there is at least one match in both tables

LEFT JOIN: Return all rows from the left table, even if there are no matches in the right table

RIGHT JOIN: Return all rows from the right table, even if there are no matches in the left table

FULL JOIN: Return rows when there is a match in one of the tables

```
mysql> select patient_id, sex, patient.age, height, weight, hip, bmi from patient inner join sample using(patient_id) limit 10;
+------------+------+------+--------+--------+------+------+
| patient_id | sex  | age  | height | weight | hip  | bmi  |
+------------+------+------+--------+--------+------+------+
|          1 |    1 |   51 |  171.5 |   91.8 | 112.5 | 31.2 |
|         10 |    1 |   51 |  169.3 |   86.4 |  99.5 | 30.1 |
|        100 |    2 |   79 |  151.7 |   77.3 |   119 | 33.6 |
|        101 |    2 |   75 |  155.5 |   77.6 | 114.9 | 32.1 |
|        101 |    2 |   75 |  155.5 |   74.6 | 114.9 | 30.9 |
|        102 |    2 |   68 |    145 |   61.8 | 105.4 | 29.4 |
|        102 |    2 |   68 |    145 |   63.8 | 102.2 | 30.3 |
|        103 |    2 |   70 |  157.4 |   75.6 | 116.5 | 30.5 |
|        103 |    2 |   70 |  157.4 |   75.4 | 113.3 | 30.4 |
|        104 |    2 |   55 |  162.3 |     89 | 122.8 | 33.8 |
+------------+------+------+--------+--------+------+------+
10 rows in set (0.00 sec)
```

Joining

```
SELECT * FROM table1, table2;

SELECT * FROM table1 INNER JOIN table2 ON table1.id=table2.id;

SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id;

SELECT * FROM table1 LEFT JOIN table2 USING (id);

SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id
  LEFT JOIN table3 ON table2.id=table3.id;
```

```
select patient_id, patient.bmi_group, round(delta_weight,2), datediff(date_v2,
date_v1), round(s1.bact/s1.firm,4),round(s2.bact/s2.firm,4) from bact2firm_gg
as s1 inner join forcompare on (sid_v1 = s1.sample_id) inner join bact2firm_gg
as s2 on (s2.sample_id = sid_v2) inner join sample on (sample.sample_id =
s2.sample_id) inner join patient using(patient_id);
```

```
select patient_id, bmi_group, s1.stat_value,s2.stat_value from sample_stat as s1
inner join forcompare on (sid_v1 = s1.sample_id) inner join sample_stat as s2 on
(s2.sample_id = sid_v2) inner join sample using(sample_id) inner join patient
using(patient_id) where s1.stat_name = 'shannon_rdp50' and s2.stat_name =
'shannon_rdp50';
```

Joining

```
table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table

table_factor:
    tbl_name [[AS] alias] [index_hint_list]
  | table_subquery [AS] alias
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
       ON conditional_expr }

join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition:
    ON conditional_expr
  | USING (column_list)

index_hint_list:
    index_hint [, index_hint] ...
```

```
index_hint:
    USE {INDEX|KEY}
      [{FOR {JOIN|ORDER BY|GROUP BY}] ([index_list])
  | IGNORE {INDEX|KEY}
      [{FOR {JOIN|ORDER BY|GROUP BY}] (index_list)
  | FORCE {INDEX|KEY}
      [{FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

index_list:
    index_name [, index_name] ...
```

MySQL Admin

```
[bcantarel@bcantarel-lx amish]$ mysqladmin -u bcantarel -p -h hannibal processlist
Enter password:
+---------+-----------+-----------------------------------------+-----------+---------+------+-------+------------------+
| Id      | User      | Host                                    | db        | Command | Time | State | Info             |
+---------+-----------+-----------------------------------------+-----------+---------+------+-------+------------------+
| 1390855 | bcantarel | bcantarel-lx.igs.umaryland.edu:44036 | cfl_amish | Sleep   | 656  |       |                  |
| 1391694 | bcantarel | bcantarel-lx.igs.umaryland.edu:52692 |           | Query   | 0    |       | show processlist |
+---------+-----------+-----------------------------------------+-----------+---------+------+-------+------------------+
```

```
mysql> explain select patient_id, sex, patient.age, height, weight, hip, bmi
    -> from patient right join sample using(patient_id) where
    -> patient.age = (select max(age) from patient);
+----+-------------+---------+------+---------------+------------+---------+-------------------------------+------+-------------+
| id | select_type | table   | type | possible_keys | key        | key_len | ref                           | rows | Extra       |
+----+-------------+---------+------+---------------+------------+---------+-------------------------------+------+-------------+
|  1 | PRIMARY     | patient | ALL  | NULL          | NULL       | NULL    | NULL                          |  391 | Using where |
|  1 | PRIMARY     | sample  | ref  | patient_id    | patient_id | 5       | cfl_amish.patient.patient_id |    1 | Using where |
|  2 | SUBQUERY    | patient | ALL  | NULL          | NULL       | NULL    | NULL                          |  391 |             |
+----+-------------+---------+------+---------------+------------+---------+-------------------------------+------+-------------+
3 rows in set (0.00 sec)
```

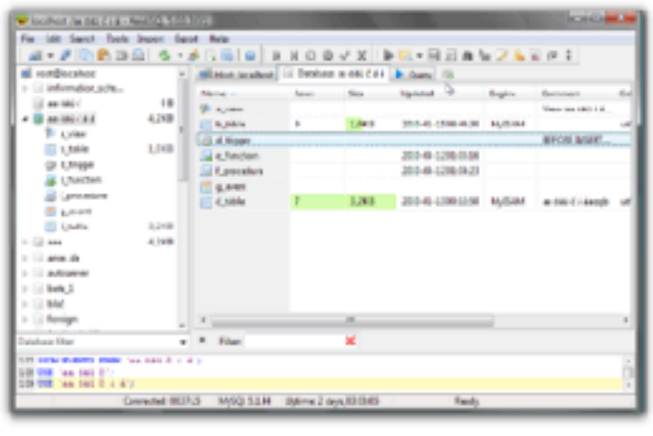
MySQL GUI
Windows

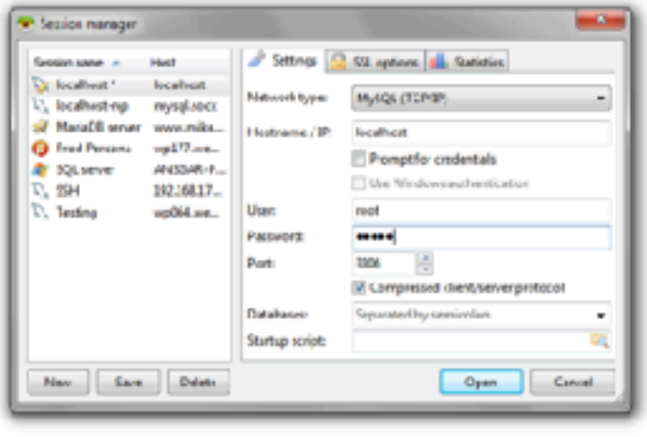
HeidiSQL

Home | Downloads | Screenshots | Forum | Donate | Bugtracker | Help

What's this?

HeidiSQL is a useful and reliable tool designed for web developers using the popular MySQL server, Microsoft SQL databases and PostgreSQL. It enables you to browse and edit data, create and edit tables, views, procedures, triggers and scheduled events. Also, you can export structure and data either to SQL file, clipboard or to other servers. ... read about features or see some screenshots.





MySQL Workbench

Download MySQL Workbench

MySQL Workbench provides DBAs and developers an integrated tools environment for:

- Database Design & Modeling
- SQL Development (replacing MySQL Query Browser)
- Database Administration (replacing MySQL Administrator)

The Community (OSS) Edition is available from this page under the GPL.

Download source packages of LGPL libraries: [+]

MySQL open source software is provided under the GPL License.

OEMs, ISVs and VARs can purchase commercial licenses.

To learn more about MySQL Workbench:

- MySQL Workbench Documentation and Change History
- Forums and Blogs

Documentation

MySQL 5.7 Reference Manual

version 5.7

Including MySQL NDB Cluster 7.5 and NDB Cluster 7.6

Abstract

This is the MySQL™ Reference Manual. It documents MySQL 5.7 through 5.7.22, as well as NDB Cluster releases based on version 7.5 of NDB through 5.7.20-ndb-7.5.9, respectively.

MySQL 5.7 features. This manual describes features that are not included in every edition of MySQL 5.7; such features may not be included in the edition of MySQL 5.7 licensed to you. If you have any questions about the features included in your edition of MySQL 5.7, refer to your MySQL 5.7 license agreement or contact your Oracle sales representative.

For notes detailing the changes in each release, see the MySQL 5.7 Release Notes.

For legal information, see the Legal Notices.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Licensing information—MySQL 5.7. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL 5.7, see this document for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL 5.7, see this document for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Licensing information—NDB Cluster. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL NDB Cluster 7.5, see this document for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL NDB Cluster 7.5, see this document for licensing information, including licensing information relating to third-party software that may be included in this Community release. If you are using MySQL NDB Cluster version 7.6, now in development, see this document for licensing information, including licensing information relating to third-party software that may be included in this Developer Preview release.

Document generated on: 2018-01-03 (revision: 55314)

https://dev.mysql.com/doc/refman/5.7/en/

sqldf : R package for running SQL statements on R data frames

```
> DF <- data.frame(a = 1:5, b = letters[1:5])
> sqldf("select * from DF")
  a b
1 1 a
2 2 b
3 3 c
4 4 d
5 5 e
> sqldf("select avg(a) mean, variance(a) var from DF”)
15
  mean var
1    3 2.5
```

sqldf : R package for running SQL statements on R data frames

```
> sqldf("select * from iris limit 5")
  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
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
> sqldf("select count(*) from iris")
  count(*)
1      150
> sqldf("select Species, count(*) from iris group by Species")
     Species count(*)
1     setosa       50
2 versicolor       50
3  virginica       50
```

Filter Data using SQL Commands

```
library(sqldf)
setwd("~/Desktop")
tbl1.filename="sample_data.csv"
tbl2.filename="table2.csv"
sgroup <- “monocytes"
sqlcommand <- paste("select * from file where SampleGroup ='",sgroup,"'",sep='')
wp.df <- read.csv.sql(tbl1.filename, sql = sqlcommand, sep = "\t")
```

```
> read.csv.sql(tbl1.filename, sql = paste("select * from file where SampleGroup ='",sgroup,"'",sep=''),sep = ",")
    SampleID      Tissue SampleGroup SubjectID     Organism     Race   SampleName Gender        FullPathToFqR1
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens    White 53_Monocytes female SRR1551069_1.fastq.gz
2 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens    White 21_Monocytes female SRR1551055_1.fastq.gz
3 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens    White 20_Monocytes female SRR1551048_1.fastq.gz
4 SRR1550987 Whole.Blood   monocytes        44 Homo sapiens Hispanic 44_Monocytes female SRR1550987_1.fastq.gz
        FullPathToFqR2
1 SRR1551069_2.fastq.gz
2 SRR1551055_2.fastq.gz
3 SRR1551048_2.fastq.gz
4 SRR1550987_2.fastq.gz
```

join sqldf

```
setwd(“~/Desktop")
tbl1 <- read.csv(file="sample_data.csv",header=TRUE)
tbl2 <- read.csv(file=“table2.csv",header=TRUE)
sqldf("select * from tbl1 inner join tbl2 using(SampleID)")
    SampleID      Tissue SampleGroup SubjectID     Organism     Race     SampleName
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens    White   53_Monocytes
2 SRR1551068 Whole.Blood neutrophils        53 Homo sapiens    White 53_Neutrophils
3 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens    White   21_Monocytes
4 SRR1551054 Whole.Blood neutrophils        21 Homo sapiens    White 21_Neutrophils
5 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens    White   20_Monocytes
6 SRR1551047 Whole.Blood neutrophils        20 Homo sapiens    White 20_Neutrophils
7 SRR1550987 Whole.Blood   monocytes        44 Homo sapiens Hispanic   44_Monocytes
8 SRR1550986 Whole.Blood neutrophils        44 Homo sapiens Hispanic 44_Neutrophils
  Gender         FullPathToFqR1         FullPathToFqR2 SubjectID BMI
1 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz         53  23
2 female SRR1551068_1.fastq.gz SRR1551068_2.fastq.gz         53  23
3 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz         21  28
4 female SRR1551054_1.fastq.gz SRR1551054_2.fastq.gz         21  28
5 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz         20  35
6 female SRR1551047_1.fastq.gz SRR1551047_2.fastq.gz         20  35
7 female SRR1550987_1.fastq.gz SRR1550987_2.fastq.gz         44  31
8 female SRR1550986_1.fastq.gz SRR1550986_2.fastq.gz         44  31
```

join sqldf

```
setwd(“~/Desktop")
tbl1 <- read.csv(file="sample_data.csv",header=TRUE)
tbl2 <- read.csv(file=“table2.csv",header=TRUE)
sqldf("select SampleID, SampleGroup,BMI from tbl1 inner join tbl2
using(SampleID)”)
```

```
    SampleID SampleGroup BMI
1 SRR1551069   monocytes  23
2 SRR1551068 neutrophils  23
3 SRR1551055   monocytes  28
4 SRR1551054 neutrophils  28
5 SRR1551048   monocytes  35
6 SRR1551047 neutrophils  35
7 SRR1550987   monocytes  31
8 SRR1550986 neutrophils  31
```

dplyr

dplyr is a package for data manipulation, that uses intuitive commands

- add new variables that are functions of existing variables
  - mutate()
- pick variables based on their names.
  - select()
- pick cases based on their values
  - filter()
- reduce multiple values down to a single summary
  - summarise()
- change the ordering of the rows
  - arrange()

dplyr: filter

```
> filter(tbl1,SampleGroup=='monocytes')
    SampleID      Tissue SampleGroup SubjectID     Organism     Race   SampleName
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens    White 53_Monocytes
2 SRR1551055 Whole.Blood   monocytes        21 Homo sapiens    White 21_Monocytes
3 SRR1551048 Whole.Blood   monocytes        20 Homo sapiens    White 20_Monocytes
4 SRR1550987 Whole.Blood   monocytes        44 Homo sapiens Hispanic 44_Monocytes
  Gender       FullPathToFqR1       FullPathToFqR2
1 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz
2 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz
3 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz
4 female SRR1550987_1.fastq.gz SRR1550987_2.fastq.gz


> filter(tbl1,SampleGroup=='monocytes',SubjectID==53)
    SampleID      Tissue SampleGroup SubjectID     Organism  Race   SampleName Gender
1 SRR1551069 Whole.Blood   monocytes        53 Homo sapiens White 53_Monocytes female
       FullPathToFqR1       FullPathToFqR2
1 SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz
```

dplyr: arrange

```
> arrange(tbl1,SampleGroup,SubjectID)
   SampleID      Tissue SampleGroup SubjectID     Organism     Race     SampleName
1 SRR1551048 Whole.Blood    monocytes        20 Homo sapiens    White   20_Monocytes
2 SRR1551055 Whole.Blood    monocytes        21 Homo sapiens    White   21_Monocytes
3 SRR1550987 Whole.Blood    monocytes        44 Homo sapiens Hispanic   44_Monocytes
4 SRR1551069 Whole.Blood    monocytes        53 Homo sapiens    White   53_Monocytes
5 SRR1551047 Whole.Blood neutrophils        20 Homo sapiens    White 20_Neutrophils
6 SRR1551054 Whole.Blood neutrophils        21 Homo sapiens    White 21_Neutrophils
7 SRR1550986 Whole.Blood neutrophils        44 Homo sapiens Hispanic 44_Neutrophils
8 SRR1551068 Whole.Blood neutrophils        53 Homo sapiens    White 53_Neutrophils
  Gender        FullPathToFqR1        FullPathToFqR2
1 female SRR1551048_1.fastq.gz SRR1551048_2.fastq.gz
2 female SRR1551055_1.fastq.gz SRR1551055_2.fastq.gz
3 female SRR1550987_1.fastq.gz SRR1550987_2.fastq.gz
4 female SRR1551069_1.fastq.gz SRR1551069_2.fastq.gz
5 female SRR1551047_1.fastq.gz SRR1551047_2.fastq.gz
6 female SRR1551054_1.fastq.gz SRR1551054_2.fastq.gz
7 female SRR1550986_1.fastq.gz SRR1550986_2.fastq.gz
8 female SRR1551068_1.fastq.gz SRR1551068_2.fastq.gz
```

dplyr: select

```
> select(tbl1,SampleGroup,SubjectID)
  SampleGroup SubjectID
1   monocytes        53
2 neutrophils        53
3   monocytes        21
4 neutrophils        21
5   monocytes        20
6 neutrophils        20
7   monocytes        44
8 neutrophils        44
```

dplyr: mutate

```
> head(tbl)
  sex  ageYear ageMonth heightIn weightLb
1   f 11.91667      143     56.3     85.0
2   f 12.91667      155     62.3    105.0
3   f 12.75000      153     63.3    108.0
4   f 13.41667      161     59.0     92.0
5   f 15.91667      191     62.5    112.5
6   f 14.25000      171     62.5    112.0
```

```
> head(mutate(tbl,weightKg=weightLb/2.2))
  sex  ageYear ageMonth heightIn weightLb weightKg
1   f 11.91667      143     56.3     85.0 38.63636
2   f 12.91667      155     62.3    105.0 47.72727
3   f 12.75000      153     63.3    108.0 49.09091
4   f 13.41667      161     59.0     92.0 41.81818
5   f 15.91667      191     62.5    112.5 51.13636
6   f 14.25000      171     62.5    112.0 50.90909
```

dplyr: summarize

```
> summarize(tbl,mean.height=mean(heightIn))
  mean.height
1    61.36456
>
summarize(group_by(tbl,sex),mean.height=mean(heightIn))
# A tibble: 2 x 2
     sex mean.height
  <fctr>       <dbl>
1      f    60.52613
2      m    62.10317
>
summarize(group_by(tbl,sex),mean.height=mean(heightIn),
mean.weight=mean(weightLb))
# A tibble: 2 x 3
     sex mean.height mean.weight
  <fctr>       <dbl>       <dbl>
1      f    60.52613    98.87838
2      m    62.10317   103.44841
```

Questions?