Advanced topics in R

Research Informatics Core

January 17, 2023

Contents

1 PART 1

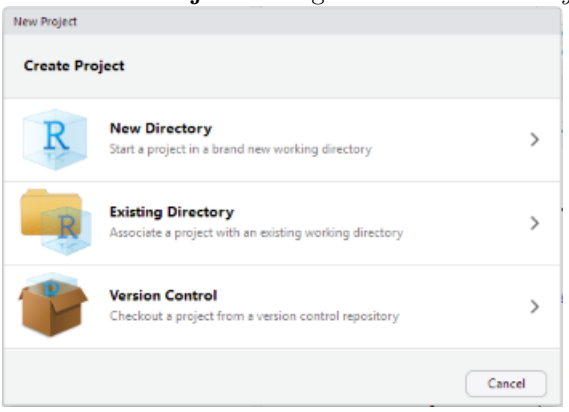
REFRESHER: Using R Studio

TIPS

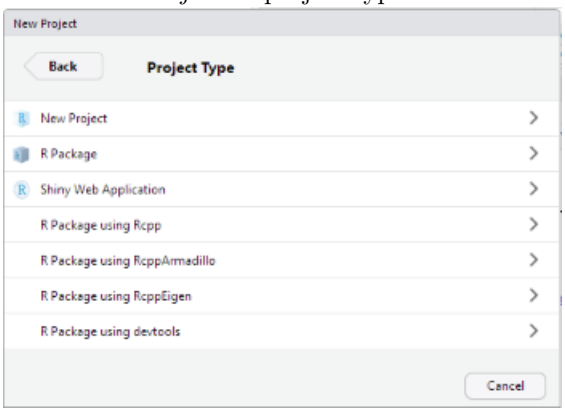
- Use Tab to auto complete
- Use up arrow to get previous command

1.0.1 Create a new project in R Studio

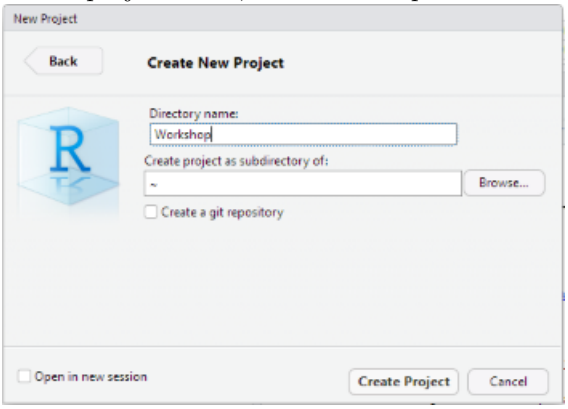
1. From *File* menu select *New Project...*
2. From **New Project** Dialog select *New Directory*



3. Select *New Project* as project type.



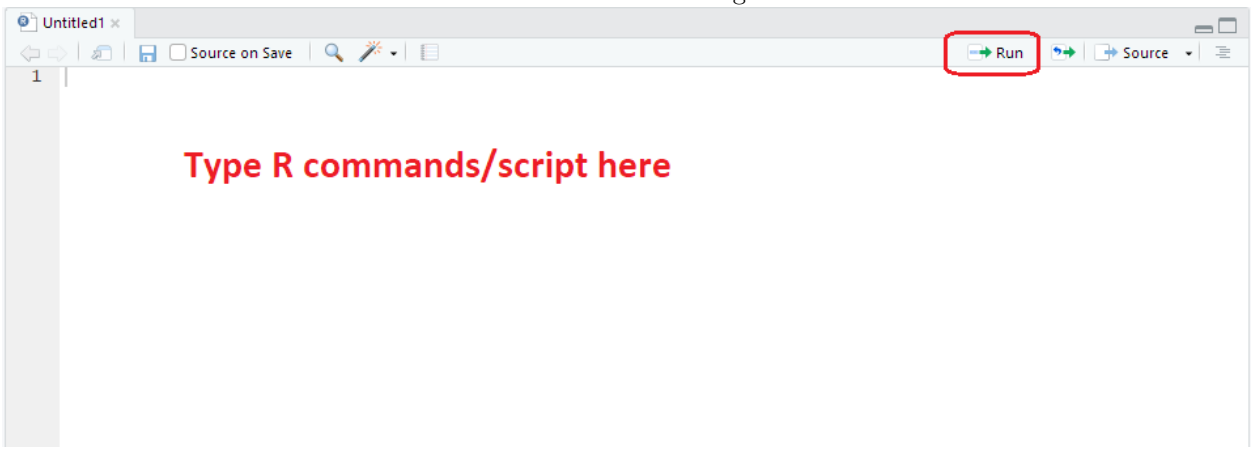
4. Give a project name, i.e. "Workshop" and click **Create Project** button.



1.0.2 Create a new script.

1. From the *File* menu select *New File* > *R Script*
2. Save file as "my_Rscript.R"

Write commands in code editor of R Studio and run them using icon **Run** in R Studio.



1.1 apply function vs for loop

We'll compute the mean expression of all genes two ways. The `apply` function is much faster than using a `for` loop.

1.1.1 Named vectors

The `apply` function returns a named vector, so here is a quick reminder on how named vectors work.

```
# create a named vector by adding state names to their respective area
names(state.area) <- state.name
# now we can access the area of a state by name
state.area["Illinois"]
```

```
## Illinois
##    56400
```

```
# or get a list of states by area
sort(state.area[state.area < state.area["Illinois"]])
```

```
##   Rhode Island       Delaware    Connecticut         Hawaii     New Jersey
##           1214           2057           5009           6450           7836
##  Massachusetts  New Hampshire        Vermont       Maryland  West Virginia
##           8257           9304           9609          10577          24181
## South Carolina          Maine        Indiana       Kentucky       Virginia
##          31055          33215          36291          40395          40815
##           Ohio      Tennessee   Pennsylvania    Mississippi      Louisiana
##          41222          42244          45333          47716          48523
##       New York        Alabama North Carolina       Arkansas      Wisconsin
##          49576          51609          52586          53104          56154
##           Iowa
##          56290
```

0. Download some example data.

```
gene.exp <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/TCGA.SKCM.20.txt",
           header=T, row.names=1)
dim(gene.exp)
```

```
## [1] 20501    20
```

```
gene.exp[1:6, 1:2]
```

```
##       TCGA.D3.A3MR.06A.11R.A21D.07 TCGA.D3.A8GI.06A.11R.A37K.07
## A1BG                      9.226257                     9.5060469
## A1CF                      0.000000                     0.0000000
## A2BP1                     0.000000                     0.8351965
## A2LD1                     5.792785                     6.2065983
## A2ML1                     1.695638                     0.0000000
## A2M                      15.092980                    13.4262673
```

1. Use `apply` function

```
# obtain mean expression for each gene using "apply" function
# the system.time() command tells us how long it took to run a command
system.time(mean.gene.exp <- apply(gene.exp, 1, mean))
```

```
##    user  system elapsed
##   0.171   0.009   0.179
```

```
head(mean.gene.exp)
```

```
##        A1BG        A1CF       A2BP1       A2LD1       A2ML1         A2M
##  7.66027247  0.00000000  0.04663362  6.39365297  1.54585053 14.59680606
```

2. Use `for` loop

```
# obtain mean expression for each gene using "for" loop
# make sure you type out the whole system.time and loop code before executing
N.genes <- nrow(gene.exp)
mean.gene.exp <- vector()
system.time(for (i in 1:N.genes){
    mean.gene.exp[i] <- mean(as.numeric(gene.exp[i, ]))
})
```

```
##    user  system elapsed
##   5.121   0.045   5.166
```

```
head(mean.gene.exp)
```

```
## [1]  7.66027247  0.00000000  0.04663362  6.39365297  1.54585053 14.59680606
```

The loop is MUCH slower, but the result is the same. However note that the `for` loop did not return a named vector.

1.2 Combining and sub-setting data

1. Use `rbind` function to combine datasets row by row

```
data1 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data1.txt", header=T)
data1
```

```
##   ID Age Sex
## 1  1  55   F
## 2  2  50   M
## 3  3  62   M
## 4  4  52   F
## 5  5  50   F
## 6  6  42   M
## 7  7  63   F
## 8  8  52   F
```

```
data2 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data2.txt", header=T)
data2
```

```
##   ID Age Sex
## 1  9  70   F
## 2 10  59   M
```

```
# use `rbind` to combine datasets row by row
data1_2 <- rbind(data1, data2)
data1_2
```

```
##    ID Age Sex
## 1   1  55   F
## 2   2  50   M
## 3   3  62   M
## 4   4  52   F
## 5   5  50   F
## 6   6  42   M
## 7   7  63   F
## 8   8  52   F
## 9   9  70   F
## 10 10  59   M
```

2. Use `cbind` function to combine datasets column by column

```
data3 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data3.txt", header=T)
data3
```

```
##   ID  Race Vital.Status
## 1  1 Asian        Alive
## 2  2 Black        Alive
## 3  3 White         Dead
## 4  4 Black        Alive
## 5  5 White         Dead
## 6  6 White        Alive
## 7  7 White         Dead
## 8  8 White        Alive
```

```r
# use `cbind` to combine datasets column by column
# Note that The subject IDs must be in the same order in the two tables.
data1_3 <- cbind(data1, data3[, 2:3])
data1_3
```

```
##   ID Age Sex  Race Vital.Status
## 1  1  55   F Asian        Alive
## 2  2  50   M Black        Alive
## 3  3  62   M White         Dead
## 4  4  52   F Black        Alive
## 5  5  50   F White         Dead
## 6  6  42   M White        Alive
## 7  7  63   F White         Dead
## 8  8  52   F White        Alive
```

3. Use `merge` function to intersect datasets

```r
data4 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data4.txt", header=T)
data4
```

```
##    ID Age Sex
## 1   1  55   F
## 2   2  50   M
## 3   3  62   M
## 4   4  52   F
## 5   5  50   F
## 6   6  42   M
## 7   7  63   F
## 8   8  52   F
## 9   9  70   F
## 10 10  59   M
## 11 11  56   F
## 12 12  74   F
## 13 13  61   M
## 14 14  39   M
## 15 15  52   F
```

```r
data5 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data5.txt", header=T)
data5
```

```
##    ID  Race Vital.Status
## 1   4 Black        Alive
## 2   5 White         Dead
## 3   6 White        Alive
## 4   1 Asian        Alive
## 5   2 Black        Alive
## 6   3 White         Dead
## 7   7 White         Dead
## 8   8 White        Alive
## 9   9 Black        Alive
## 10 10 Black        Alive
## 11 11 Black        Alive
## 12 12 Asian         Dead
```

```r
# use `merge` to obtain intersection of datasets
# The subject IDs don't need to be in the same order.
data.intersection <- merge(data4, data5, by="ID")
data.intersection
```

```
##    ID Age Sex  Race Vital.Status
## 1   1  55   F Asian        Alive
## 2   2  50   M Black        Alive
## 3   3  62   M White         Dead
## 4   4  52   F Black        Alive
## 5   5  50   F White         Dead
## 6   6  42   M White        Alive
## 7   7  63   F White         Dead
## 8   8  52   F White        Alive
## 9   9  70   F Black        Alive
## 10 10  59   M Black        Alive
## 11 11  56   F Black        Alive
## 12 12  74   F Asian         Dead
```

4. Use `merge` function to make union of datasets

```r
# use `merge` to obtain union of datasets with `all=T`
# The subject IDs don't need to be in the same order.
data.union <- merge(data4, data5, by="ID", all=T)
data.union
```

```
##    ID Age Sex  Race Vital.Status
## 1   1  55   F Asian        Alive
## 2   2  50   M Black        Alive
## 3   3  62   M White         Dead
## 4   4  52   F Black        Alive
## 5   5  50   F White         Dead
## 6   6  42   M White        Alive
## 7   7  63   F White         Dead
## 8   8  52   F White        Alive
## 9   9  70   F Black        Alive
## 10 10  59   M Black        Alive
## 11 11  56   F Black        Alive
## 12 12  74   F Asian         Dead
## 13 13  61   M  <NA>         <NA>
## 14 14  39   M  <NA>         <NA>
## 15 15  52   F  <NA>         <NA>
```

5. What if we have multiple matches for the same ID?

```r
# list of visits per patient
data6 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data6.txt", header=T)
data6
```

```
##     Visit ID
## 1  Visit1  1
## 2  Visit1  2
## 3  Visit1  3
## 4  Visit1  4
## 5  Visit1  5
## 6  Visit1  6
## 7  Visit1  7
```

```
## 8  Visit1  8
## 9  Visit1  9
## 10 Visit1 10
## 11 Visit1 11
## 12 Visit1 12
## 13 Visit1 13
## 14 Visit1 14
## 15 Visit1 15
## 16 Visit2  1
## 17 Visit2  2
## 18 Visit2  3
## 19 Visit2  4
## 20 Visit2  5
## 21 Visit2  6
## 22 Visit2  7
## 23 Visit2  8
## 24 Visit2  9
## 25 Visit2 10
## 26 Visit2 11
## 27 Visit2 12
## 28 Visit2 13
## 29 Visit2 14
## 30 Visit2 15
```

```
# check the number of replicates for each "ID" in data6
table(data6$ID)
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
##  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
```

```
# compare that with the number of replicate for each "ID" in data4
table(data4$ID)
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

```
# merge data6 and data4 by "ID"
data.visits <- merge(data6, data4, by="ID")
data.visits
```

```
##    ID  Visit Age Sex
## 1   1 Visit1  55   F
## 2   1 Visit2  55   F
## 3   2 Visit1  50   M
## 4   2 Visit2  50   M
## 5   3 Visit1  62   M
## 6   3 Visit2  62   M
## 7   4 Visit2  52   F
## 8   4 Visit1  52   F
## 9   5 Visit1  50   F
## 10  5 Visit2  50   F
## 11  6 Visit1  42   M
## 12  6 Visit2  42   M
## 13  7 Visit1  63   F
## 14  7 Visit2  63   F
```

```
## 15  8 Visit2  52   F
## 16  8 Visit1  52   F
## 17  9 Visit1  70   F
## 18  9 Visit2  70   F
## 19 10 Visit1  59   M
## 20 10 Visit2  59   M
## 21 11 Visit1  56   F
## 22 11 Visit2  56   F
## 23 12 Visit2  74   F
## 24 12 Visit1  74   F
## 25 13 Visit1  61   M
## 26 13 Visit2  61   M
## 27 14 Visit1  39   M
## 28 14 Visit2  39   M
## 29 15 Visit1  52   F
## 30 15 Visit2  52   F
```

6. Add a new column to an existing data frame

```
# Add a new column "Agecat" (age category)
data.union$Agecat[data.union$Age > 60] <- "Old"
data.union
```

```
##    ID Age Sex  Race Vital.Status Agecat
## 1   1  55   F Asian        Alive   <NA>
## 2   2  50   M Black        Alive   <NA>
## 3   3  62   M White         Dead    Old
## 4   4  52   F Black        Alive   <NA>
## 5   5  50   F White         Dead   <NA>
## 6   6  42   M White        Alive   <NA>
## 7   7  63   F White         Dead    Old
## 8   8  52   F White        Alive   <NA>
## 9   9  70   F Black        Alive    Old
## 10 10  59   M Black        Alive   <NA>
## 11 11  56   F Black        Alive   <NA>
## 12 12  74   F Asian         Dead    Old
## 13 13  61   M  <NA>         <NA>    Old
## 14 14  39   M  <NA>         <NA>   <NA>
## 15 15  52   F  <NA>         <NA>   <NA>
```

```
data.union$Agecat[data.union$Age <= 60] <- "Young"
data.union
```

```
##    ID Age Sex  Race Vital.Status Agecat
## 1   1  55   F Asian        Alive  Young
## 2   2  50   M Black        Alive  Young
## 3   3  62   M White         Dead    Old
## 4   4  52   F Black        Alive  Young
## 5   5  50   F White         Dead  Young
## 6   6  42   M White        Alive  Young
## 7   7  63   F White         Dead    Old
## 8   8  52   F White        Alive  Young
## 9   9  70   F Black        Alive    Old
## 10 10  59   M Black        Alive  Young
## 11 11  56   F Black        Alive  Young
## 12 12  74   F Asian         Dead    Old
```

```
## 13 13  61   M  <NA>         <NA>    Old
## 14 14  39   M  <NA>         <NA>  Young
## 15 15  52   F  <NA>         <NA>  Young
```

7. Use `subset` function to select subset of data

```
# select subset of data (young and female)
data.sub <- subset(data.union, Agecat=="Young" & Sex=="F")
data.sub
```

```
##    ID Age Sex  Race Vital.Status Agecat
## 1   1  55   F Asian        Alive  Young
## 4   4  52   F Black        Alive  Young
## 5   5  50   F White         Dead  Young
## 8   8  52   F White        Alive  Young
## 11 11  56   F Black        Alive  Young
## 15 15  52   F  <NA>         <NA>  Young
```

1.3 String substitution and grep

1. Use sub or gsub function to substitute string

```
sampleIDs <- colnames(read.table("https://uic-ric.github.io/workshop-data/advanced_R/TCGA.small.txt",
            header=T, row.names=1))
sampleIDs
```

```
## [1] "TCGA.D3.A3MR.06A.11R.A21D.07" "TCGA.D3.A8GI.06A.11R.A37K.07"
## [3] "TCGA.D9.A3Z1.06A.11R.A239.07" "TCGA.EE.A2GM.06B.11R.A18S.07"
```

1. Use sub or gsub function to substitute string

```
# `sub` only substitutes the first match.
# "fixed=T": string to be matched as it is
sub(".", "-", sampleIDs, fixed=T)
```

```
## [1] "TCGA-D3.A3MR.06A.11R.A21D.07" "TCGA-D3.A8GI.06A.11R.A37K.07"
## [3] "TCGA-D9.A3Z1.06A.11R.A239.07" "TCGA-EE.A2GM.06B.11R.A18S.07"
```

```
# `gsub` substitutes all matches.
gsub(".", "-", sampleIDs, fixed=T)
```

```
## [1] "TCGA-D3-A3MR-06A-11R-A21D-07" "TCGA-D3-A8GI-06A-11R-A37K-07"
## [3] "TCGA-D9-A3Z1-06A-11R-A239-07" "TCGA-EE-A2GM-06B-11R-A18S-07"
```

```
# remove argument "fixed=T": string to be matched as a regular expression
# "." as a regular expression means any character
gsub(".", "-", sampleIDs)
```

```
## [1] "----------------------------" "----------------------------"
## [3] "----------------------------" "----------------------------"
```

2. Substitute string using a regular expression

```
# use a regular expression:
# "\\.[0-9]+" means to match "." following with one or more digits
gsub("\\.[0-9]+", "???", sampleIDs)
```

```
## [1] "TCGA.D3.A3MR???A???R.A21D???" "TCGA.D3.A8GI???A???R.A37K???"
## [3] "TCGA.D9.A3Z1???A???R.A239???" "TCGA.EE.A2GM???B???R.A18S???"
```

3. Use grep to search string

```
# search for sample IDs containing "06A", `grep` will return the indices
grep("06A", sampleIDs)
```

```
## [1] 1 2 3
```

```
# obtain values using the indices returned by `grep`
sampleIDs[grep("06A", sampleIDs)]
```

```
## [1] "TCGA.D3.A3MR.06A.11R.A21D.07" "TCGA.D3.A8GI.06A.11R.A37K.07"
## [3] "TCGA.D9.A3Z1.06A.11R.A239.07"
```

```
# use "value=T" to obtain values directly
grep("06A", sampleIDs, value=T)
```

```
## [1] "TCGA.D3.A3MR.06A.11R.A21D.07" "TCGA.D3.A8GI.06A.11R.A37K.07"
## [3] "TCGA.D9.A3Z1.06A.11R.A239.07"
```

4. Use `grep` to search string using a regular expression

```
# use a regular expression:
# "D[0-9]+" means to match "D" following with one or more digits
grep("D[0-9]+", sampleIDs, value=T)
```

```
## [1] "TCGA.D3.A3MR.06A.11R.A21D.07" "TCGA.D3.A8GI.06A.11R.A37K.07"
## [3] "TCGA.D9.A3Z1.06A.11R.A239.07"
```

5. Use grep to filter a data frame

```
# just print the rows with subjects in their 50s: match 5 plus any number
data4 <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/data4.txt", header=T)
data4[grep("5[0-9]",data4$Age),]
```

```
##    ID Age Sex
## 1   1  55   F
## 2   2  50   M
## 4   4  52   F
## 5   5  50   F
## 8   8  52   F
## 10 10  59   M
## 11 11  56   F
## 15 15  52   F
```

1.4 Clean messy tables

```
patients <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/clinical.data.txt",
            header=T, sep="\t")
# preview the first 15 rows
patients[1:15, ]
```

```
##         PatientID   Cohort  Age Gender Disease  Genotype
## 1   TMA-22MR-4-KP      UIC 75.9   Male       0        MT
## 2   TMA-22MR-5-KP      UIC 53.8      F       0    Mutant
## 3   TMA-23EO-6-KP      UIC 53.1   Male       1 Wild-type
## 4   TMA-23EO-7-KP      UIC 88.8      M       0        WT
## 5   TMA-24MP-8-KP      UIC 74.8      M       0        mt
## 6   TMA-24MP-9-KP Uchicago 51.7      M       0        MT
## 7   TMA-26MS-1-KP      UIC 82.5      F       1    Mutant
## 8   TMA-26MS-2-KP      UIC 55.6      M       1        WT
## 9   TMA-29CA-5-KP       NW   NA      M       1        MT
## 10 TMA-32SR-10-KP Uchicago 59.8      M                MT
## 11  TMA-32SR-1-KP Uchicago 59.2      M       1    Mutant
## 12  TMA-33MJ-5-KP      UIC 73.1      F              <NA>
## 13  TMA-33MJ-6-KP      UIC 55.2      F       .  Wildtype
## 14  TMA-34LV-2-KP      UIC 59.4      M       1  Wildtype
## 15  TMA-34LV-3-KP Uchicago 59.0      F       0 Wild type
```

1. Fix "PatientID"

```
# fix patient IDs (i.e. rownames of data frame)
# remove "-KP" at the end; "$" means to match at the end of string
patients$PatientID <- sub("-KP$", "", patients$PatientID)
head(patients)
```

```
##    PatientID   Cohort  Age Gender Disease  Genotype
## 1 TMA-22MR-4      UIC 75.9   Male       0        MT
## 2 TMA-22MR-5      UIC 53.8      F       0    Mutant
## 3 TMA-23EO-6      UIC 53.1   Male       1 Wild-type
## 4 TMA-23EO-7      UIC 88.8      M       0        WT
## 5 TMA-24MP-8      UIC 74.8      M       0        mt
## 6 TMA-24MP-9 Uchicago 51.7      M       0        MT
```

```
# replace "-" with "."
patients$PatientID <- gsub("-", ".", patients$PatientID, fixed=T)
head(patients)
```

```
##    PatientID   Cohort  Age Gender Disease  Genotype
## 1 TMA.22MR.4      UIC 75.9   Male       0        MT
## 2 TMA.22MR.5      UIC 53.8      F       0    Mutant
## 3 TMA.23EO.6      UIC 53.1   Male       1 Wild-type
## 4 TMA.23EO.7      UIC 88.8      M       0        WT
## 5 TMA.24MP.8      UIC 74.8      M       0        mt
## 6 TMA.24MP.9 Uchicago 51.7      M       0        MT
```

2. Fix "Age"

```
# Age: Change missing values with the average age in all patients.
# preview the first 10 rows
patients[1:10, ]
```

```
##      PatientID   Cohort  Age Gender Disease  Genotype
## 1   TMA.22MR.4      UIC 75.9   Male       0        MT
## 2   TMA.22MR.5      UIC 53.8      F       0    Mutant
## 3   TMA.23EO.6      UIC 53.1   Male       1 Wild-type
## 4   TMA.23EO.7      UIC 88.8      M       0        WT
## 5   TMA.24MP.8      UIC 74.8      M       0        mt
## 6   TMA.24MP.9 Uchicago 51.7      M       0        MT
## 7   TMA.26MS.1      UIC 82.5      F       1    Mutant
## 8   TMA.26MS.2      UIC 55.6      M       1        WT
## 9   TMA.29CA.5       NW   NA      M       1        MT
## 10 TMA.32SR.10 Uchicago 59.8      M                MT
```

```
# is.na() will test which elements are missing values.
# na.rm=T: missing values (NA) should be stripped before the computation proceeds.
patients$Age[is.na(patients$Age)] <- mean(patients$Age, na.rm=T)
patients[1:10, ]
```

```
##      PatientID   Cohort      Age Gender Disease  Genotype
## 1   TMA.22MR.4      UIC 75.90000   Male       0        MT
## 2   TMA.22MR.5      UIC 53.80000      F       0    Mutant
## 3   TMA.23EO.6      UIC 53.10000   Male       1 Wild-type
## 4   TMA.23EO.7      UIC 88.80000      M       0        WT
## 5   TMA.24MP.8      UIC 74.80000      M       0        mt
## 6   TMA.24MP.9 Uchicago 51.70000      M       0        MT
## 7   TMA.26MS.1      UIC 82.50000      F       1    Mutant
## 8   TMA.26MS.2      UIC 55.60000      M       1        WT
## 9   TMA.29CA.5       NW 64.34231      M       1        MT
## 10 TMA.32SR.10 Uchicago 59.80000      M                MT
```

3. Fix “Gender”

```
# check unique names for gender
unique(patients$Gender)
```

```
## [1] "Male"   "F"      "M"      "Female"
```

```
# the table() command will give us the counts
table(patients$Gender)
```

```
##
##      F Female      M   Male
##     27      1     24      3
```

```
# change "Female" to "F"; change "Male" to "M"
patients$Gender <- factor(patients$Gender)
levels(patients$Gender)
```

```
## [1] "F"      "Female" "M"      "Male"
```

```
levels(patients$Gender)[2] <- "F"
levels(patients$Gender)
```

```
## [1] "F"    "M"    "Male"
```

```
levels(patients$Gender)[3] <- "M"
levels(patients$Gender)
```

```
## [1] "F" "M"
```

```
unique(patients$Gender)
```

```
## [1] M F
## Levels: F M
```

```
# preview the first 10 rows
patients[1:10, ]
```

```
##      PatientID   Cohort      Age Gender Disease  Genotype
## 1   TMA.22MR.4      UIC 75.90000      M       0        MT
## 2   TMA.22MR.5      UIC 53.80000      F       0    Mutant
## 3   TMA.23EO.6      UIC 53.10000      M       1 Wild-type
## 4   TMA.23EO.7      UIC 88.80000      M       0        WT
## 5   TMA.24MP.8      UIC 74.80000      M       0        mt
## 6   TMA.24MP.9 Uchicago 51.70000      M       0        MT
## 7   TMA.26MS.1      UIC 82.50000      F       1    Mutant
## 8   TMA.26MS.2      UIC 55.60000      M       1        WT
## 9   TMA.29CA.5       NW 64.34231      M       1        MT
## 10 TMA.32SR.10 Uchicago 59.80000      M                MT
```

4. Fix “Disease”

```
# change "." or "" to NA (missing value)
# preview the first 15 rows
patients[1:15, ]
```

```
##      PatientID   Cohort      Age Gender Disease  Genotype
## 1   TMA.22MR.4      UIC 75.90000      M       0        MT
## 2   TMA.22MR.5      UIC 53.80000      F       0    Mutant
## 3   TMA.23EO.6      UIC 53.10000      M       1 Wild-type
```

```
## 4   TMA.23EO.7      UIC 88.80000      M       0        WT
## 5   TMA.24MP.8      UIC 74.80000      M       0        mt
## 6   TMA.24MP.9 Uchicago 51.70000      M       0        MT
## 7   TMA.26MS.1      UIC 82.50000      F       1    Mutant
## 8   TMA.26MS.2      UIC 55.60000      M       1        WT
## 9   TMA.29CA.5       NW 64.34231      M       1        MT
## 10 TMA.32SR.10 Uchicago 59.80000      M                MT
## 11  TMA.32SR.1 Uchicago 59.20000      M       1    Mutant
## 12  TMA.33MJ.5      UIC 73.10000      F              <NA>
## 13  TMA.33MJ.6      UIC 55.20000      F       .  Wildtype
## 14  TMA.34LV.2      UIC 59.40000      M       1  Wildtype
## 15  TMA.34LV.3 Uchicago 59.00000      F       0 Wild type
```

```
unique(patients$Disease)
```

```
## [1] "0" "1" ""  "."
```

```
patients$Disease[patients$Disease == "."] <- NA
patients$Disease[patients$Disease == ""] <- NA
patients$Disease <- factor(patients$Disease, levels=c("0", "1"))
unique(patients$Disease)
```

```
## [1] 0    1    <NA>
## Levels: 0 1
```

```
patients[1:15, ]
```

```
##      PatientID   Cohort      Age Gender Disease  Genotype
## 1   TMA.22MR.4      UIC 75.90000      M       0        MT
## 2   TMA.22MR.5      UIC 53.80000      F       0    Mutant
## 3   TMA.23EO.6      UIC 53.10000      M       1 Wild-type
## 4   TMA.23EO.7      UIC 88.80000      M       0        WT
## 5   TMA.24MP.8      UIC 74.80000      M       0        mt
## 6   TMA.24MP.9 Uchicago 51.70000      M       0        MT
## 7   TMA.26MS.1      UIC 82.50000      F       1    Mutant
## 8   TMA.26MS.2      UIC 55.60000      M       1        WT
## 9   TMA.29CA.5       NW 64.34231      M       1        MT
## 10 TMA.32SR.10 Uchicago 59.80000      M    <NA>        MT
## 11  TMA.32SR.1 Uchicago 59.20000      M       1    Mutant
## 12  TMA.33MJ.5      UIC 73.10000      F    <NA>      <NA>
## 13  TMA.33MJ.6      UIC 55.20000      F    <NA>  Wildtype
## 14  TMA.34LV.2      UIC 59.40000      M       1  Wildtype
## 15  TMA.34LV.3 Uchicago 59.00000      F       0 Wild type
```

5. Fix "Genotype"

```
unique(patients$Genotype)
```

```
## [1] "MT"         "Mutant"     "Wild-type" "WT"         "mt"         NA
## [7] "Wildtype"   "Wild type"
```

```
# change any string starting with "w" (or "W") to "WT";
# change any string starting with "m" (or "M") to "MT"
# ^ means to match at the beginning of string
patients$Genotype <- sub("^w.*", "WT", patients$Genotype, ignore.case=T)
patients$Genotype <- sub("^m.*", "MT", patients$Genotype, ignore.case=T)
patients$Genotype <- factor(patients$Genotype)
unique(patients$Genotype)
```

```
## [1] MT   WT   <NA>
## Levels: MT WT
```

```
# preview the first 10 rows
patients[1:10, ]
```

```
##      PatientID   Cohort      Age Gender Disease Genotype
## 1   TMA.22MR.4      UIC 75.90000      M       0       MT
## 2   TMA.22MR.5      UIC 53.80000      F       0       MT
## 3   TMA.23EO.6      UIC 53.10000      M       1       WT
## 4   TMA.23EO.7      UIC 88.80000      M       0       WT
## 5   TMA.24MP.8      UIC 74.80000      M       0       MT
## 6   TMA.24MP.9 Uchicago 51.70000      M       0       MT
## 7   TMA.26MS.1      UIC 82.50000      F       1       MT
## 8   TMA.26MS.2      UIC 55.60000      M       1       WT
## 9   TMA.29CA.5       NW 64.34231      M       1       MT
## 10 TMA.32SR.10 Uchicago 59.80000      M    <NA>       MT
```

```
# write the cleaned data into a file.
# remove the index column using row.names=F; output missing values (NA) as empty string
write.table(patients, file="clean.clinical.data.txt", quote=F, sep = "\t",
            row.names=F, na="")
```

2 PART 2

2.1 ggplot2 - Scatter plots

NOTE: If you have not previously installed `ggplot2` then install from CRAN.

```
install.packages("ggplot2")
```

Once the `ggplot2` package is installed make sure to load for this exercise

```
# load the package
library(ggplot2)
```

Read in the data for this exerise. This is a table of cell statistics from a single-cell RNA-seq project. The columns in this table are:

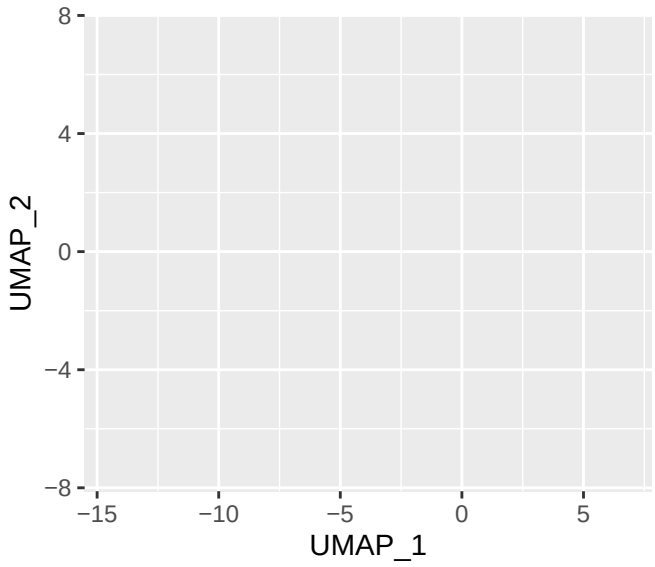
- UMIs: total UMI (unique read) count per cell.
- Genes: number of genes expressed.
- Genotype: WT or KO, 2 genotypes profiled in this experiment.
- Batch: A or B, 2 replicate captures collected in this experiment.
- PercentMT: Fraction of reads mapping to mitochondrial genes. Higher levels can indicate issues with cell viability. For this data set, cells with % MT > 15% have already been filtered out.
- HasTCR: Whether or not this cell expressed a TCR, based on a separate TCR library collected for these samples.
- Cluster: cluster assignment from analysis.
- UMAP_1 and UMAP_2: UMAP coordinates for each cell.
- Log-scaled normalized gene expression levels for several genes of interest.

```
sc <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/scRNA_cells.txt",
        header=T,row.names=1,sep="\t")
```

Scatterplots: we will make various versions of a UMAP plot.

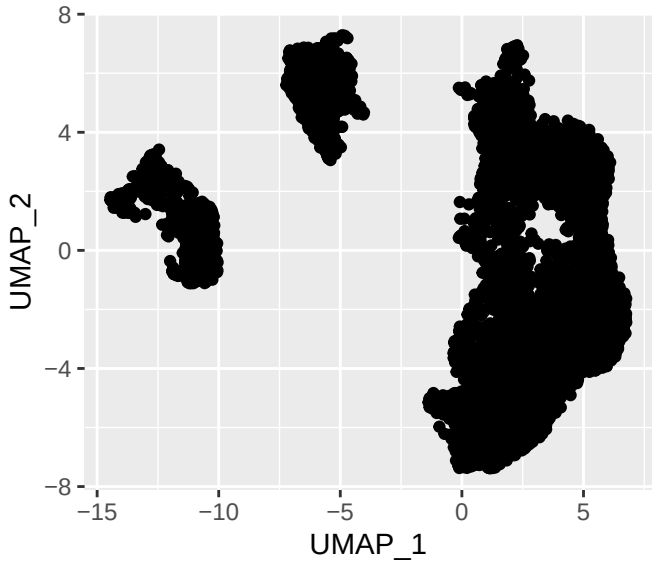
1. Create a basic frame

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2))
```



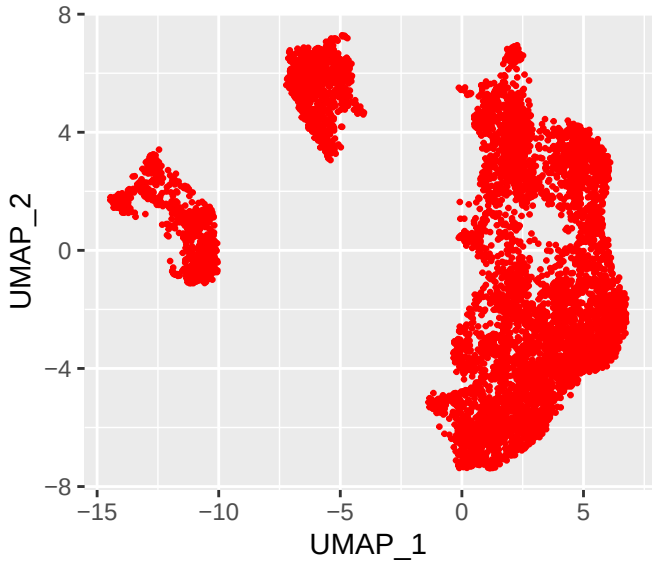
2. Make a scatterplot of the UMAP coordinates

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2)) + geom_point()
```



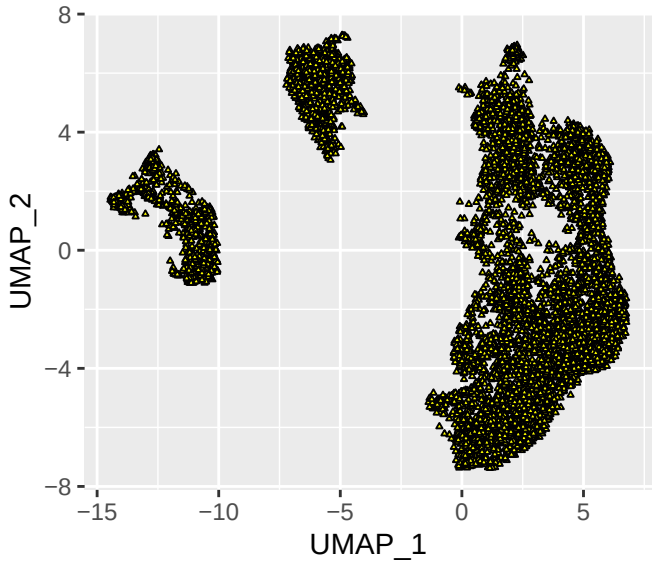
3. Add size and color

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2)) + geom_point(size=0.5, color="red")
```



4. Plot with a different shape, color with fill instead of “color”

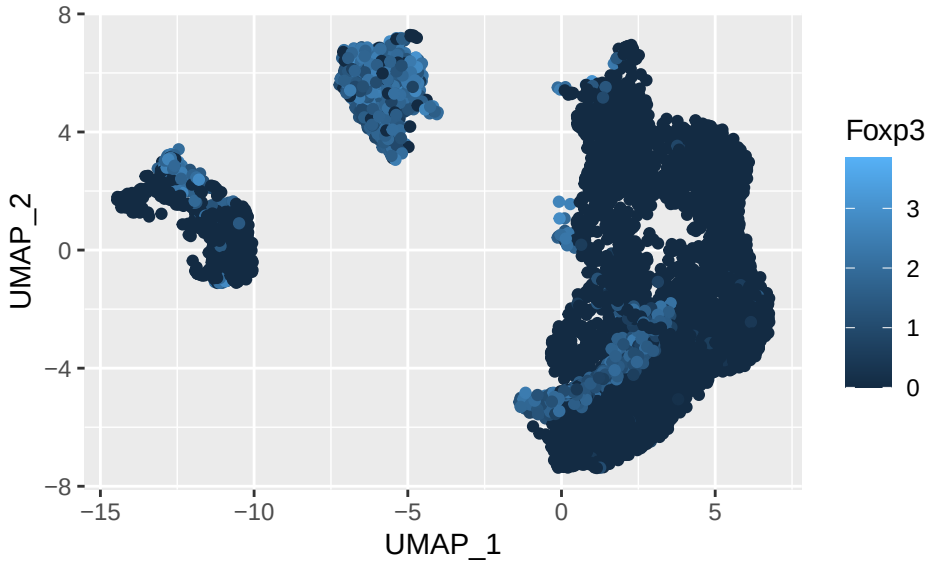
```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2)) +
  geom_point(shape=24, size=0.5, fill="yellow")
```



5. Plot with a variable color

Based on gene expression of a gene of interest:

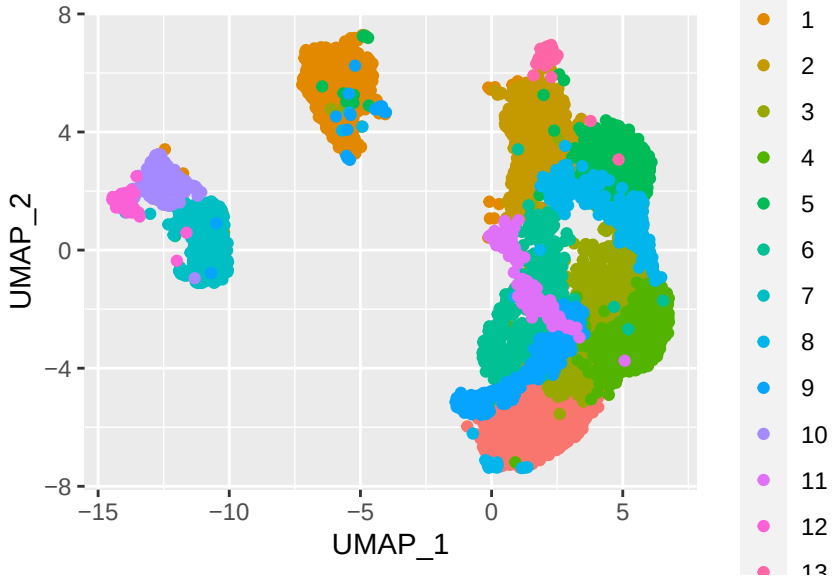
```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = Foxp3)) +
  geom_point()
```



Based on cluster:

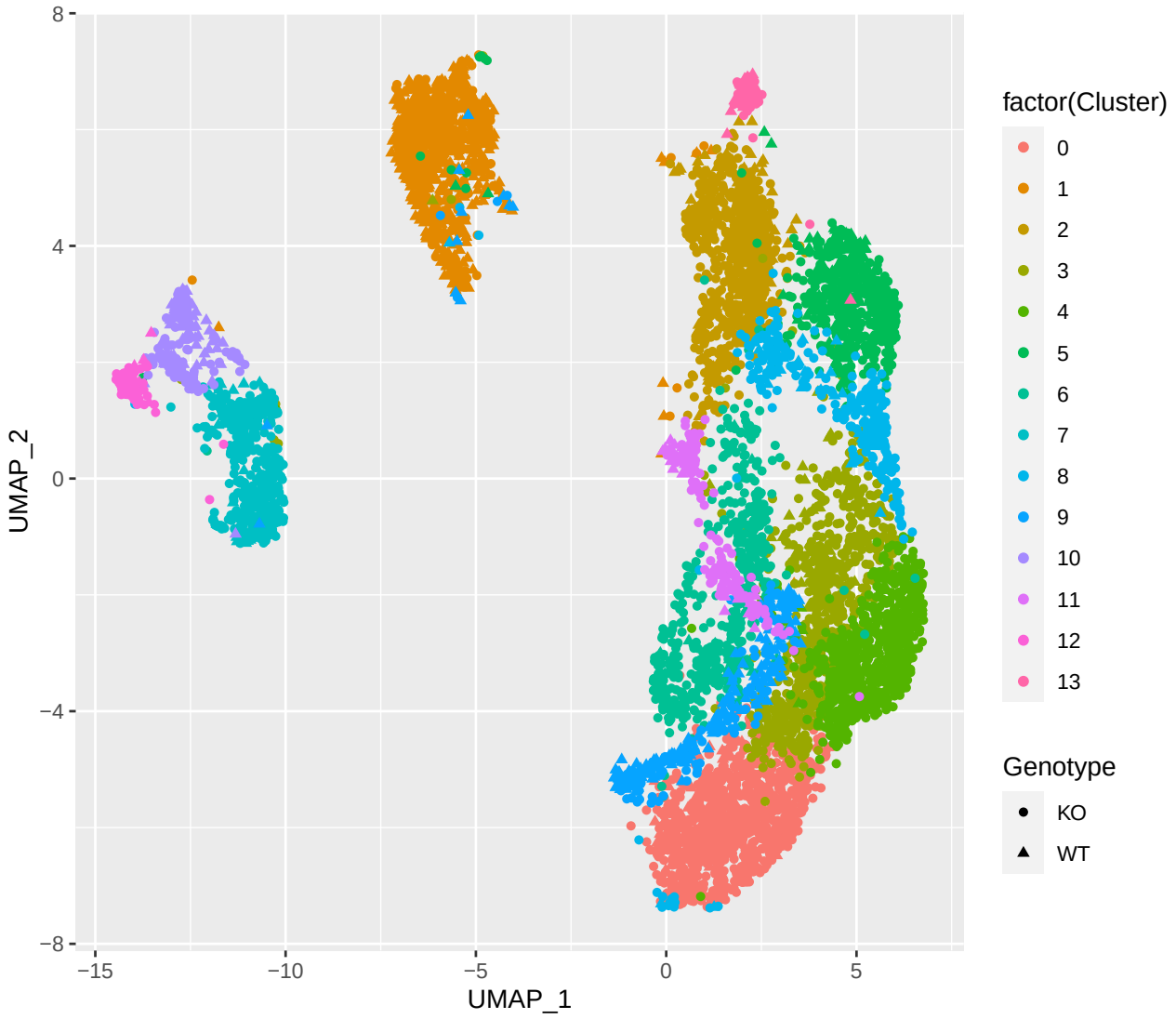
Note that clusters are numbers, so we use factor() to tell ggplot to treat this as a categorical variable.

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster))) +
  geom_point()
```



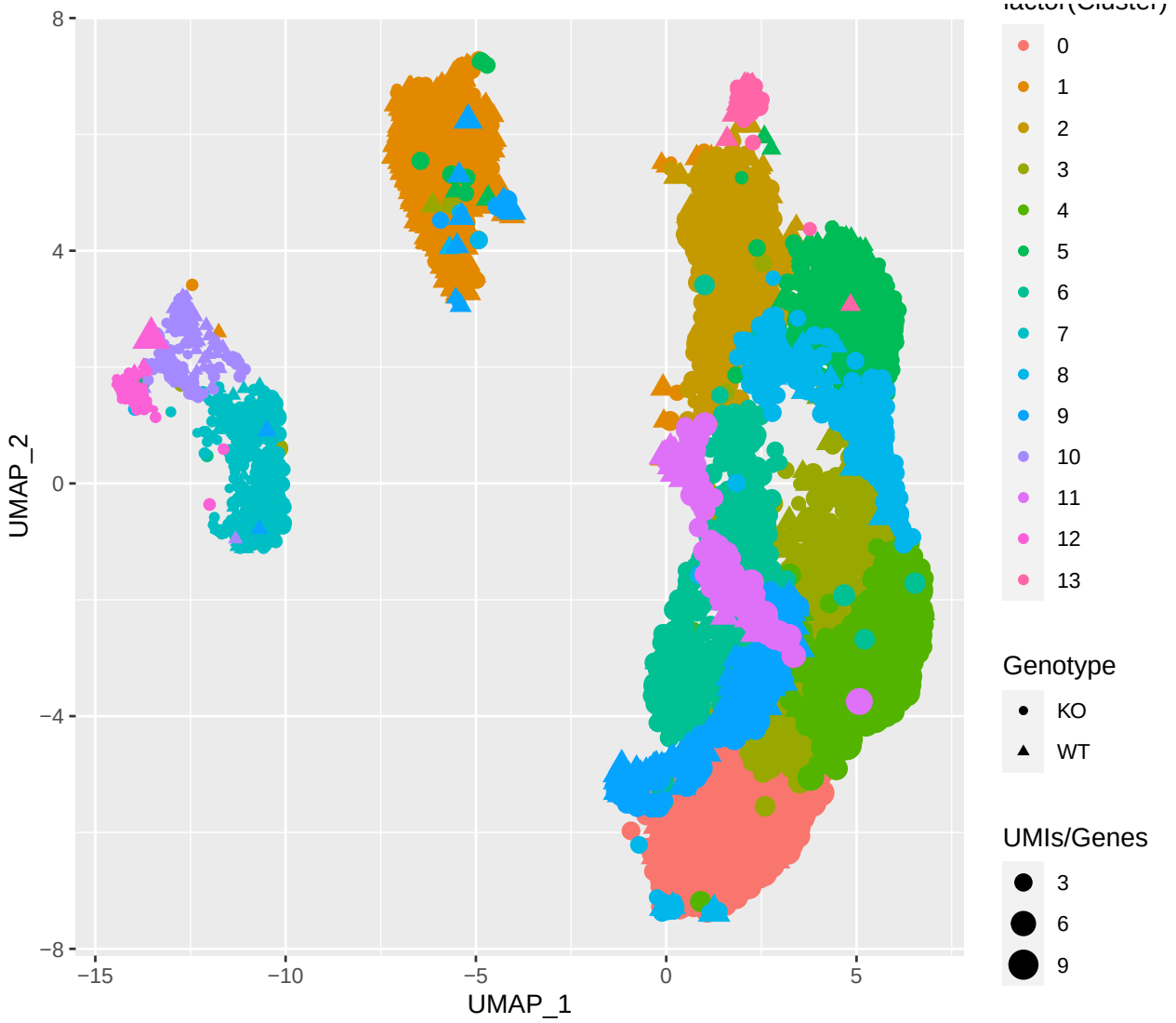
6. Also plot with a variable shape based on genotype

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster), shape = Genotype)) +
  geom_point()
```



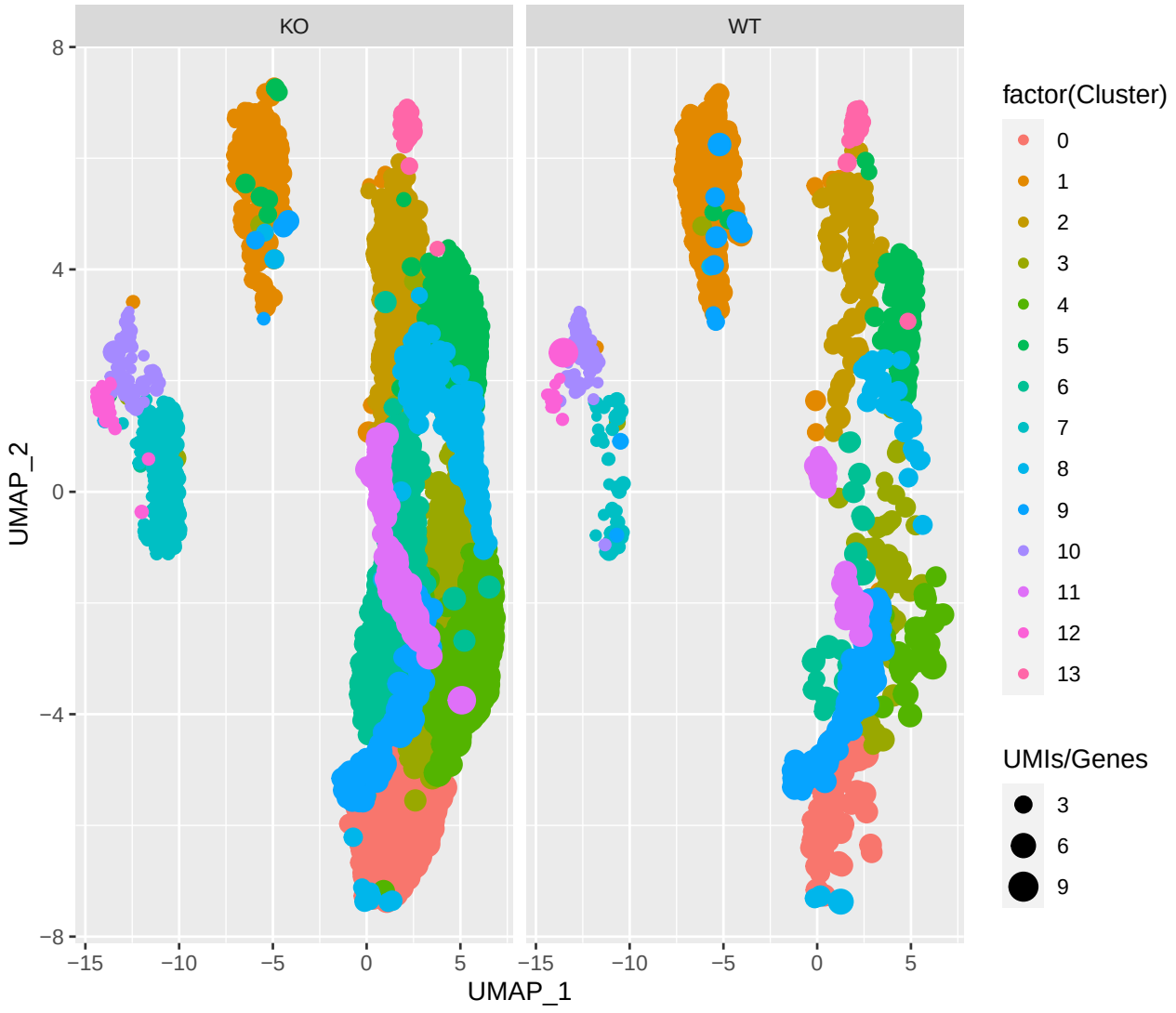
7. Also plot with a variable size based on the ratio of UMI and gene counts

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster), shape = Genotype,
  size=UMIs/Genes)) + geom_point()
```



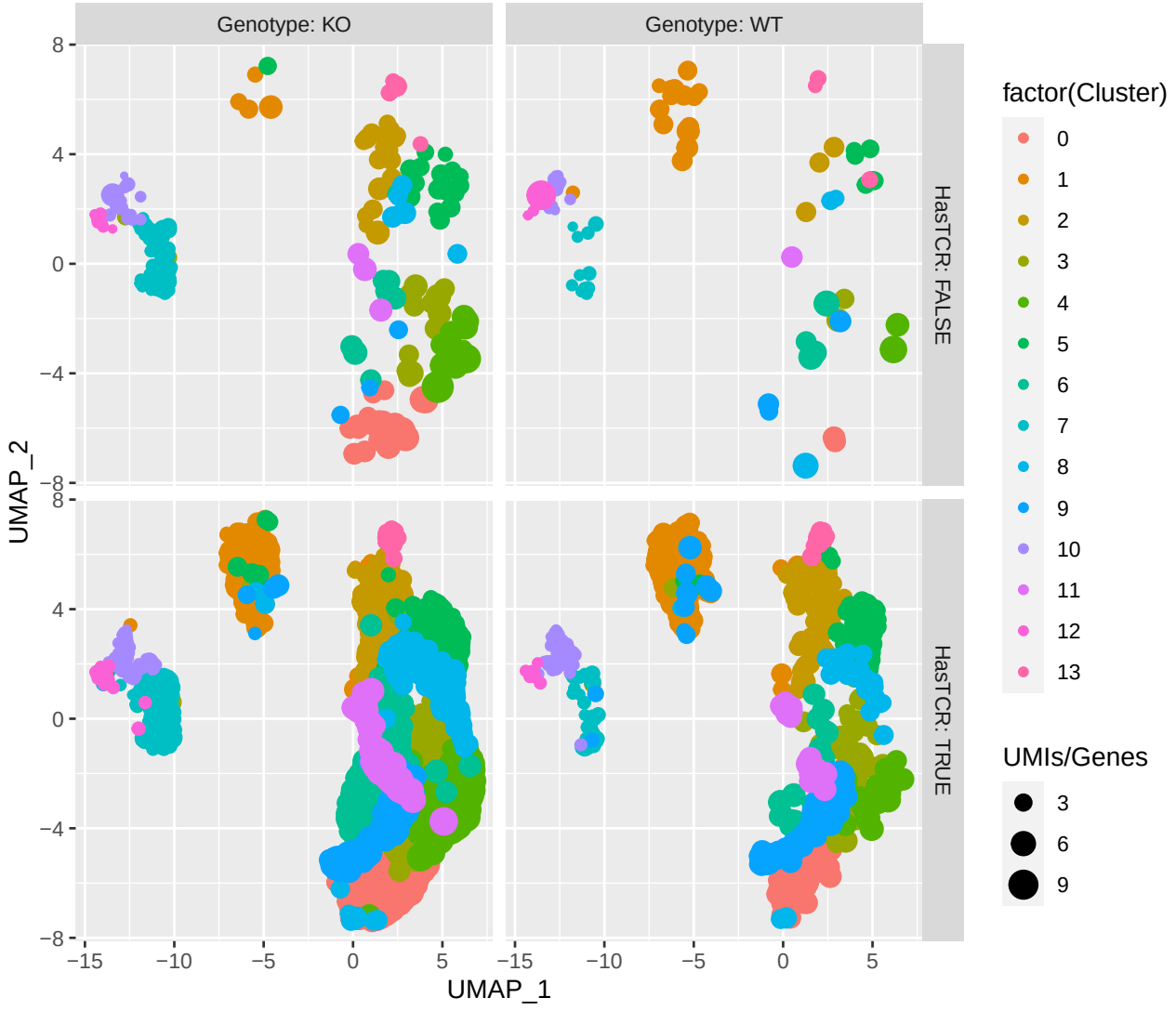
8. Facet/split of the plot by genotype instead of plotting by shape

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster), size=UMIs/Genes)) +
  geom_point() + facet_grid( ~ Genotype)
```



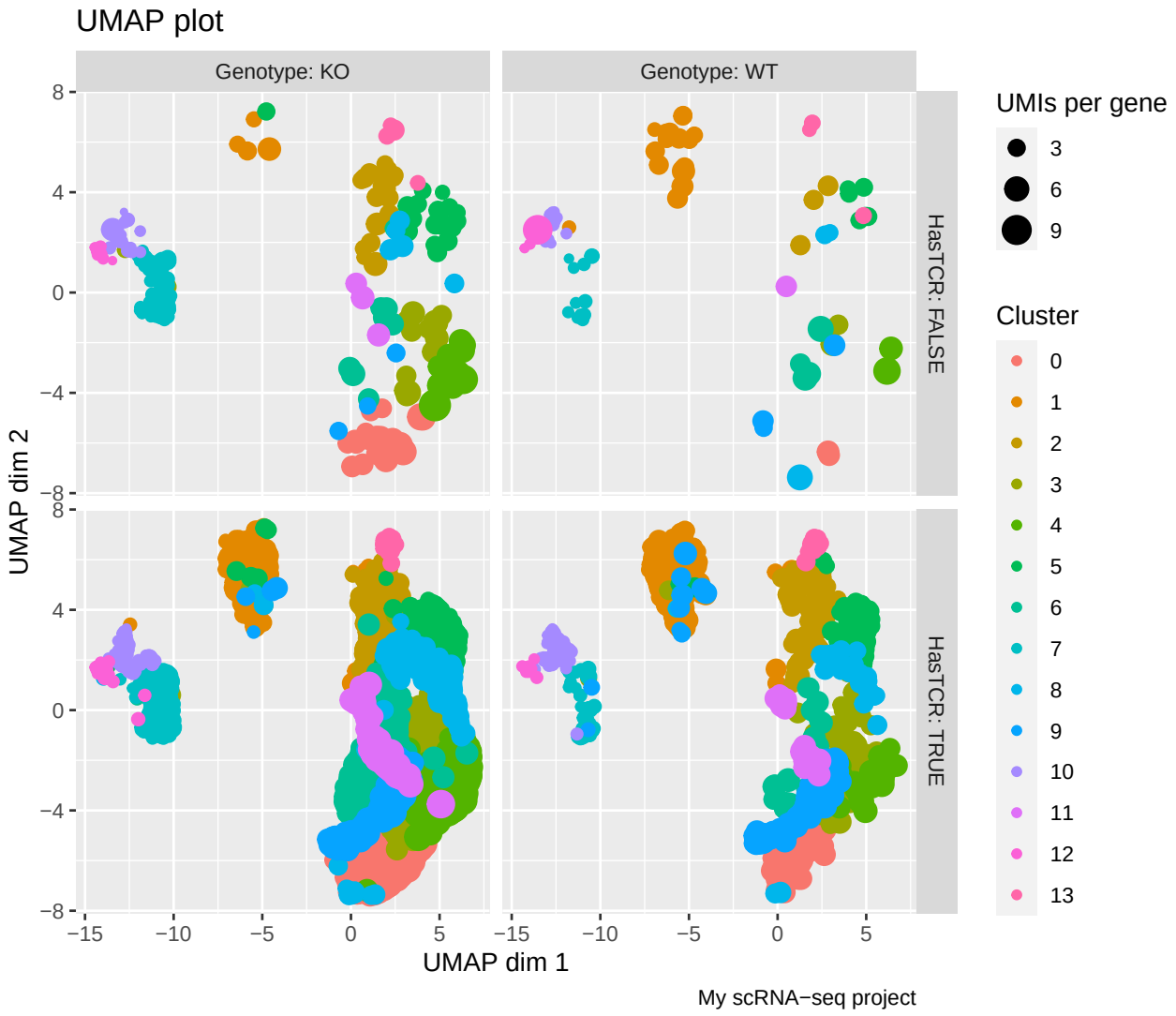
9. Facet/split of the plot by both genotype and TCR expression and better annotate the facets

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster), size=UMIs/Genes)) +
  geom_point() + facet_grid(HasTCR ~ Genotype, labeller = label_both)
```



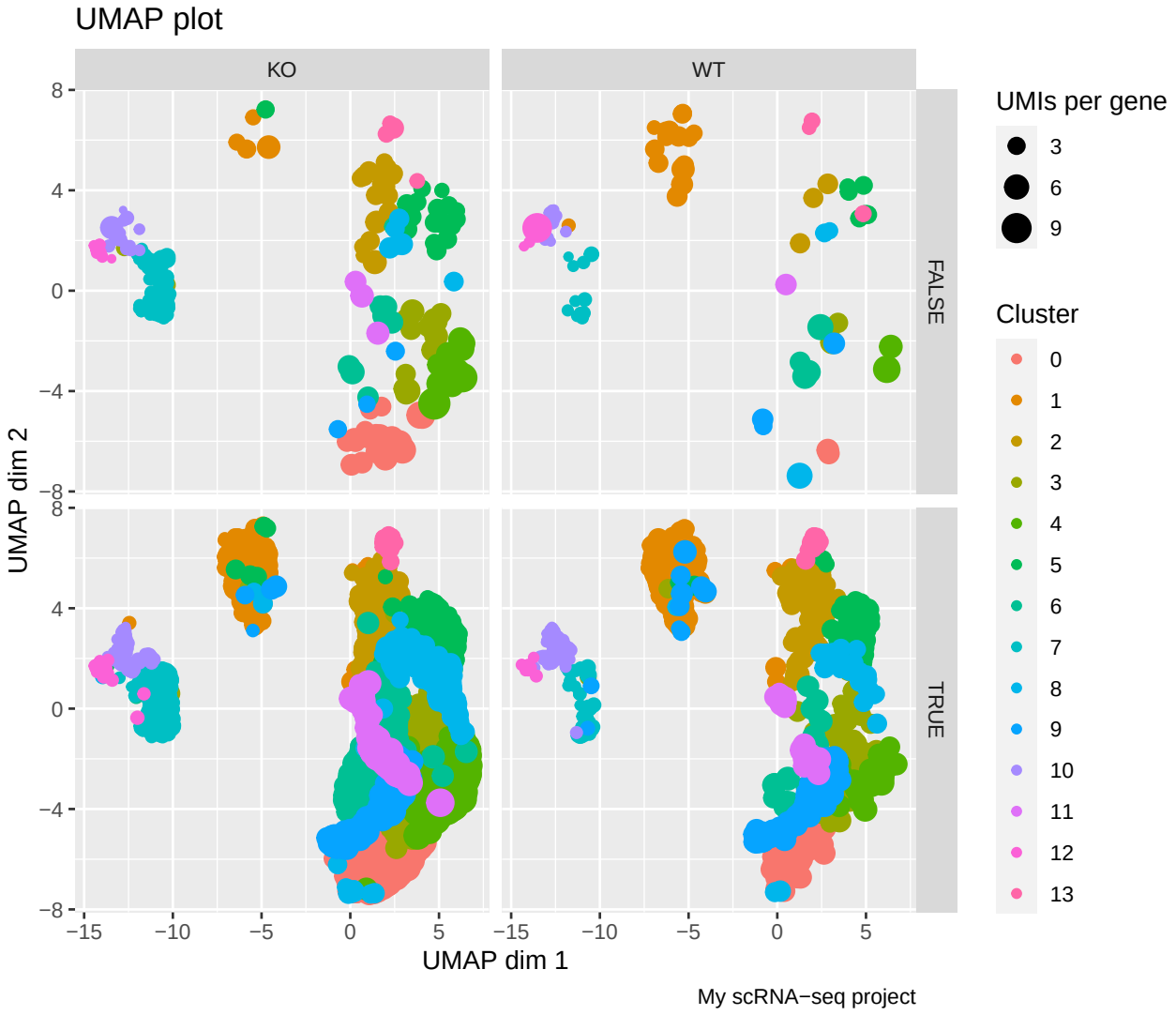
10. Update labels

```
ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster), size=UMIs/Genes)) +
  geom_point() + facet_grid(HasTCR ~ Genotype, labeller = label_both) +
  labs(title="UMAP plot", caption="My scRNA-seq project",
       x="UMAP dim 1", y="UMAP dim 2",
       color="Cluster", size="UMIs per gene")
```



11. Create the plot using variables

```
basePlot <- ggplot(sc, aes(x = UMAP_1, y = UMAP_2, color = factor(Cluster), size=UMIs/Genes))

basePlotTitle <- labs(title="UMAP plot", caption="My scRNA-seq project",
                      x="UMAP dim 1", y="UMAP dim 2",
                      color="Cluster", size="UMIs per gene")

basePlot + geom_point() + facet_grid(HasTCR ~ Genotype) + basePlotTitle
```



2.2 ggplot2 - Box and violin plot

If you have closed RStudio, make sure to reload the `ggplot2` package.

```
# load the package
library(ggplot2)
```

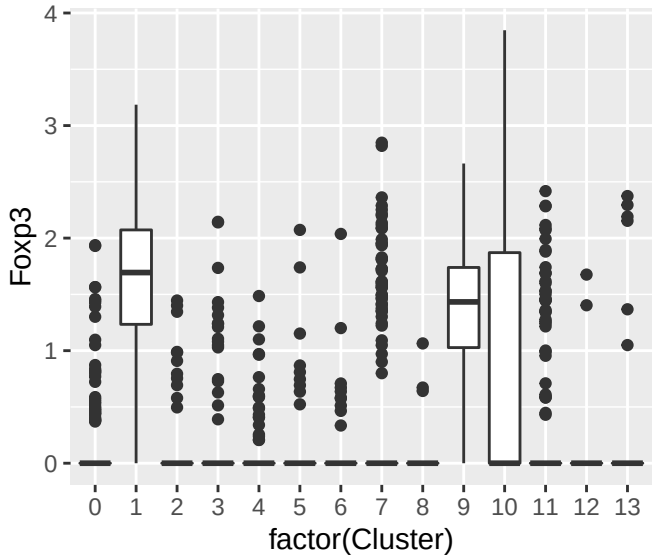
We will do some comparisons of the Foxp3 levels across different clusters.

1. Create a basic frame

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3))
```

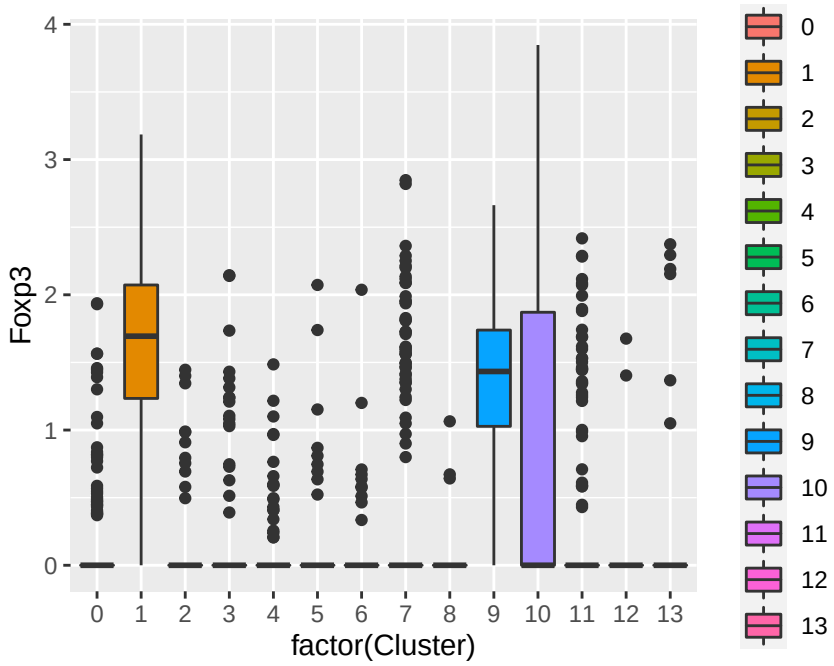
2. Add a box plot

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + geom_boxplot()
```



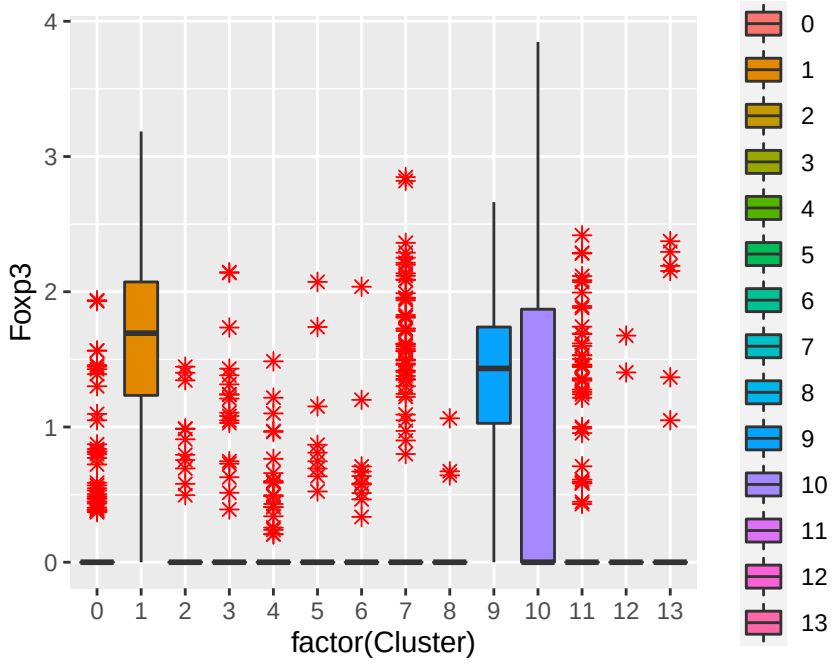
3. Set the fill color, which we can also do in the boxplot aesthetics

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + geom_boxplot(aes(fill=factor(Cluster)))
```



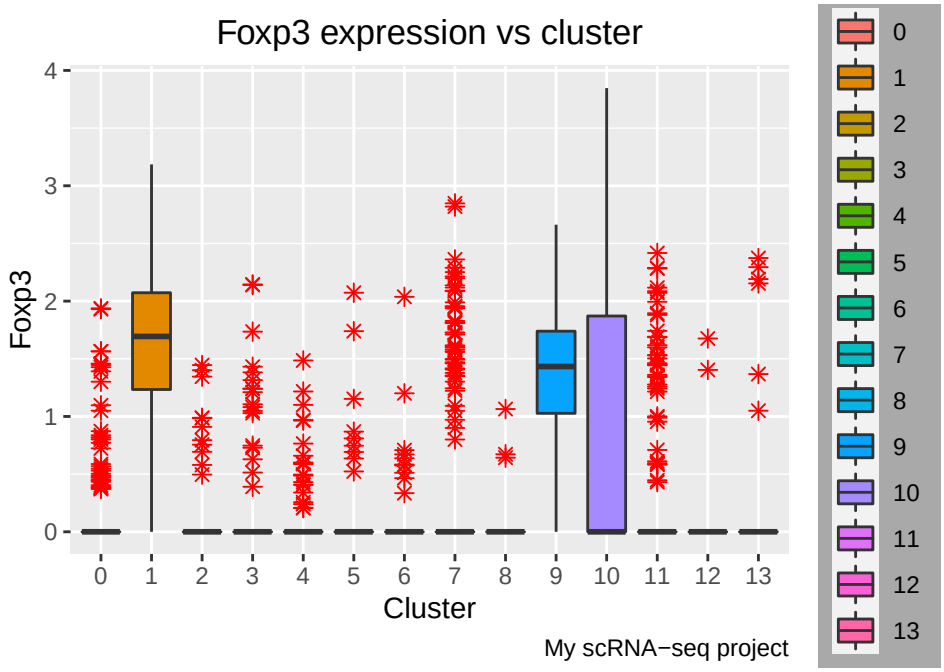
4. Set the custom color and shape for outlier points

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) +
  geom_boxplot(aes(fill=factor(Cluster)),
               outlier.fill="red", outlier.color="red",
               outlier.shape=8, outlier.size=2)
```



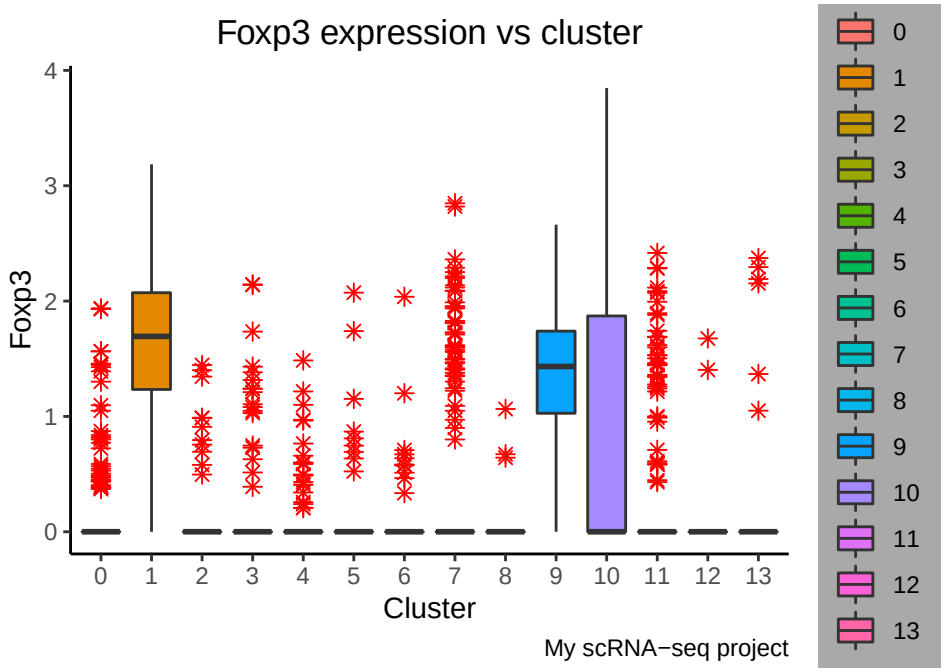
5. Create the plot using variables

```
boxParam <- geom_boxplot(aes(fill=factor(Cluster)), outlier.fill="red",
                         outlier.color="red", outlier.shape=8, outlier.size=2)

boxTitle <- labs(title="Foxp3 expression vs cluster",
                 y="Foxp3", x="Cluster",
                 caption="My scRNA-seq project", fill="Cluster")

boxTheme <- theme(plot.title = element_text(hjust=0.5),
                  legend.background = element_rect(fill="darkgray"))

ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + boxParam +
  boxTitle + boxTheme
```



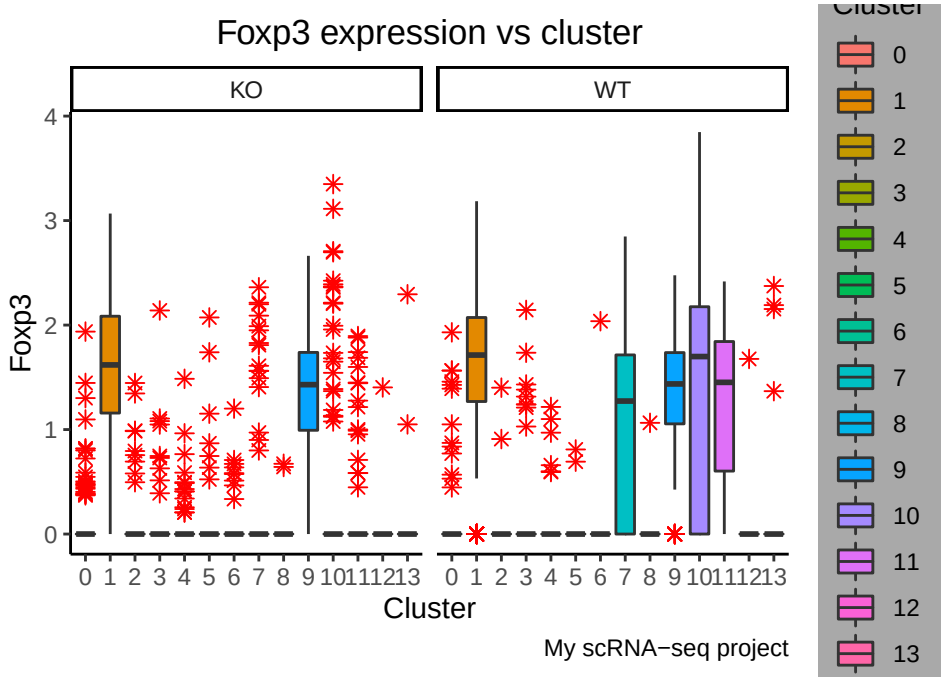
6. Change the theme

```
boxTheme <- theme_classic() + boxTheme

ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + boxParam +
  boxTitle + boxTheme
```



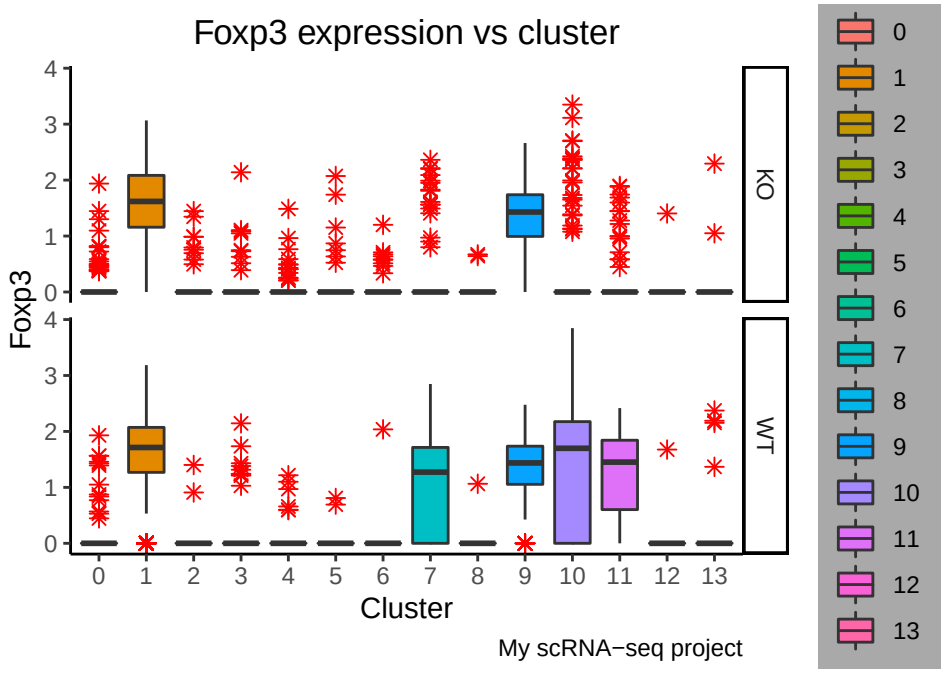
7. Add a facet

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + boxParam +
  boxTitle + boxTheme + facet_grid( ~ Genotype)
```



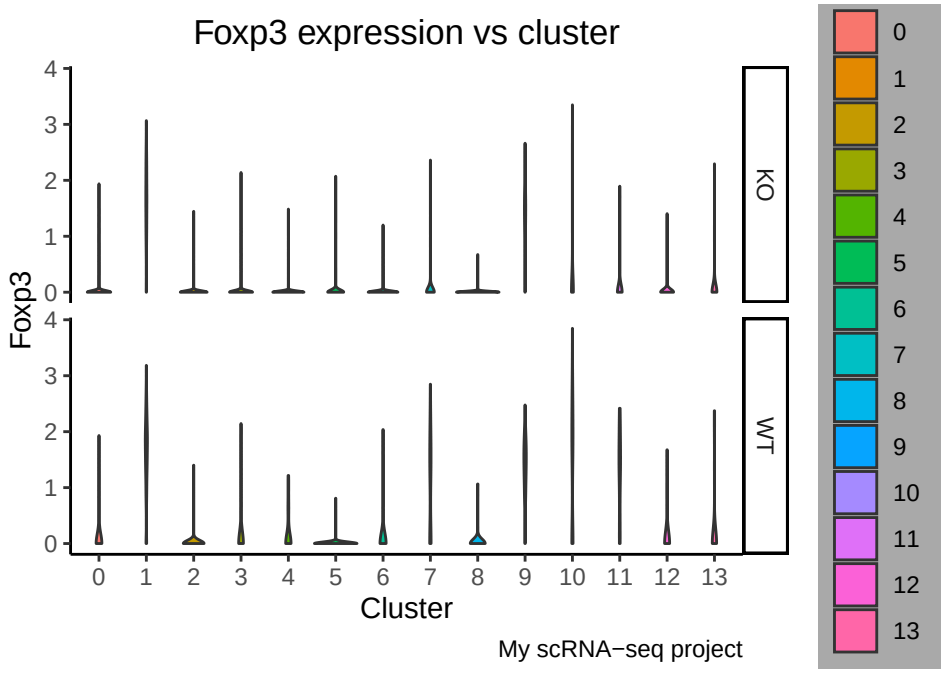
8. Adjust the facets to stack on top of each other

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + boxParam +
  boxTitle + boxTheme + facet_grid(rows=vars(Genotype))
```



9. Change to violin plot

```
ggplot(sc, aes(x=factor(Cluster), y=Foxp3)) + geom_violin(aes(fill=factor(Cluster))) +
  boxTitle + boxTheme + facet_grid(rows=vars(Genotype))
```



10. A bit more advanced: violin plot for all genes

First we need to make a "long" format of gene expression levels vs cluster.

```
# make a new data frame with just the clusters, and the gene expression levels
gene.expr <- sc[,c(7,10:17)]
head(gene.expr)
```

```
##       Cluster Foxp3      Ccr2     Cxcr6     Tcf7     Ccr7 Cxcr5     Mki67
## cell1       0     0 0.4796434 0.0000000 0.000000 0.000000     0 1.4054819
## cell2       0     0 0.0000000 0.9494970 1.092095 2.555914     0 0.3336843
## cell3       0     0 0.0000000 0.4524427 0.000000 0.000000     0 1.9053081
## cell4       0     0 0.9542333 0.9542333 0.000000 0.000000     0 2.1953778
## cell5       0     0 0.9711419 0.7390628 0.000000 0.000000     0 1.3177292
## cell6       0     0 0.0000000 1.5057177 0.000000 0.000000     0 2.0250317
##          Ccnb2
## cell1 1.546070
## cell2 1.216873
## cell3 1.718684
## cell4 1.756139
## cell5 1.574619
## cell6 1.683475
```

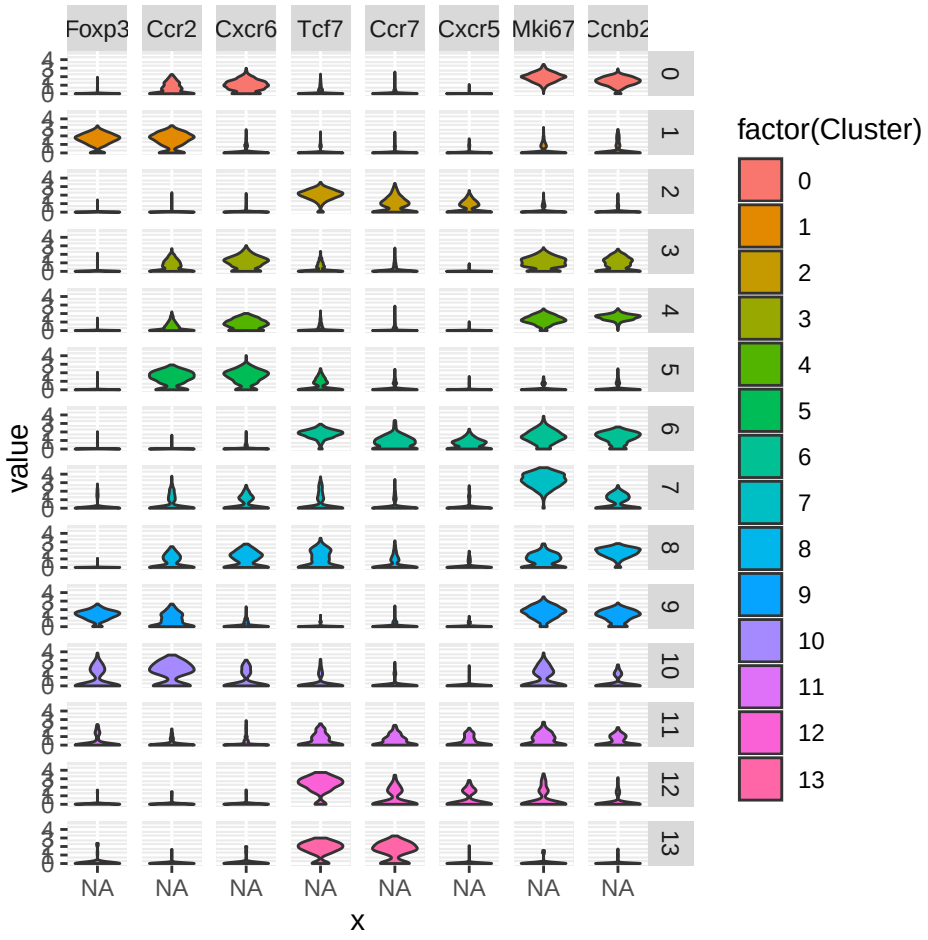
```
# convert to the "long" format. Load the reshape2 library if you haven't already.
library(reshape2)
gene.expr.long <- melt(gene.expr, id="Cluster")
head(gene.expr.long)
```

```
##   Cluster variable value
## 1       0    Foxp3     0
## 2       0    Foxp3     0
## 3       0    Foxp3     0
## 4       0    Foxp3     0
## 5       0    Foxp3     0
## 6       0    Foxp3     0
```

Plot all of the genes at once.

Note: x aesthetic is NA. We're making one plot per panel, but then a grid of panels based on cluster and gene ("variable" column).

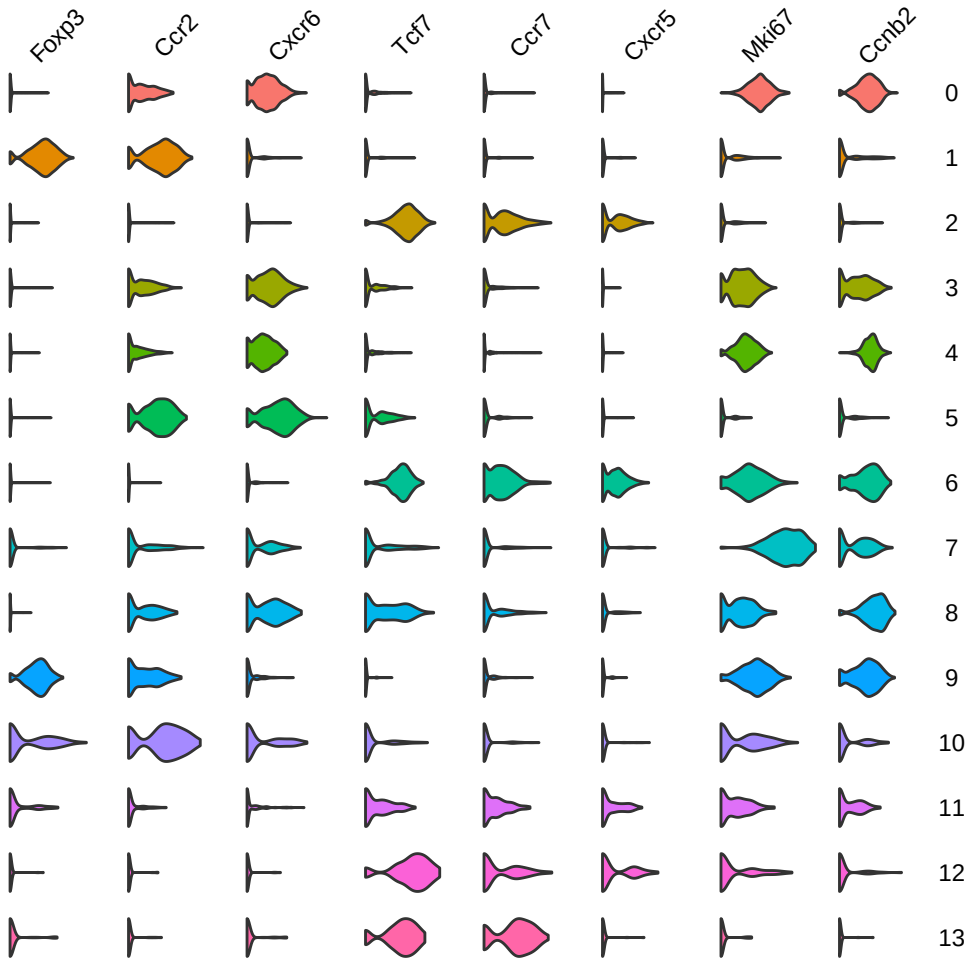
```
ggplot(gene.expr.long, aes(x=NA,y=value,fill=factor(Cluster))) +
  geom_violin() + facet_grid(Cluster ~ variable)
```



Make it prettier:

- Use “void” theme to remove all boxes and tic marks
- Flip coordinates
- Remove the legend
- Rotate the gene names

```
ggplot(gene.expr.long, aes(x=NA,y=value,fill=factor(Cluster))) +
  geom_violin() + facet_grid(Cluster ~ variable) +
  theme_void() + coord_flip() +
  theme(legend.position = "none") +
  theme(strip.text.x = element_text(angle = 45))
```



2.3 ggplot2 - Bar plots

If you have closed RStudio, make sure to reload the `ggplot2` package.

```
# load the package
library(ggplot2)
```

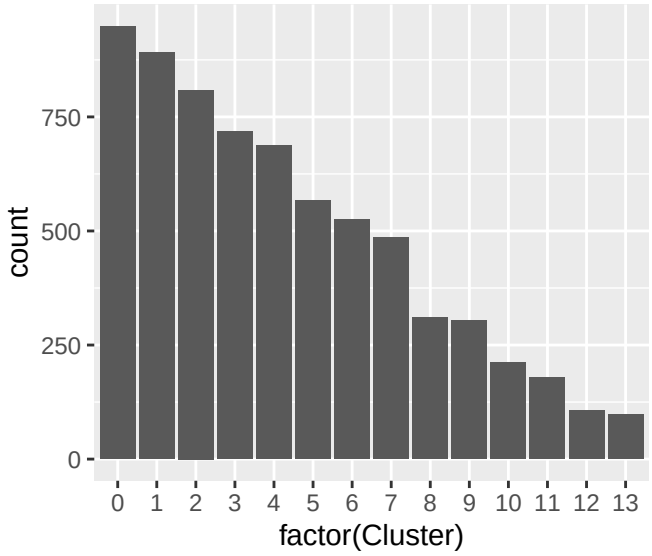
NOTE: `geom_bar()` will make a barplot by based on the number of cases in each group, so it will count all of the cells per cluster for us. `geom_col()` will make a barplot just based on values provided in a data frame.

We'll look at the number of cells in each cluster.

1. Create a basic barplot of the number of cells in each cluster

Omit the y aesthetic, as geom_bar() will count the number of entries for each value in x.

```
ggplot(sc,aes(x=factor(Cluster))) + geom_bar()
```



Alternatively, we could count the cells first and plot the counts:

```
cluster.counts <- table(sc$Cluster)
cluster.counts
```

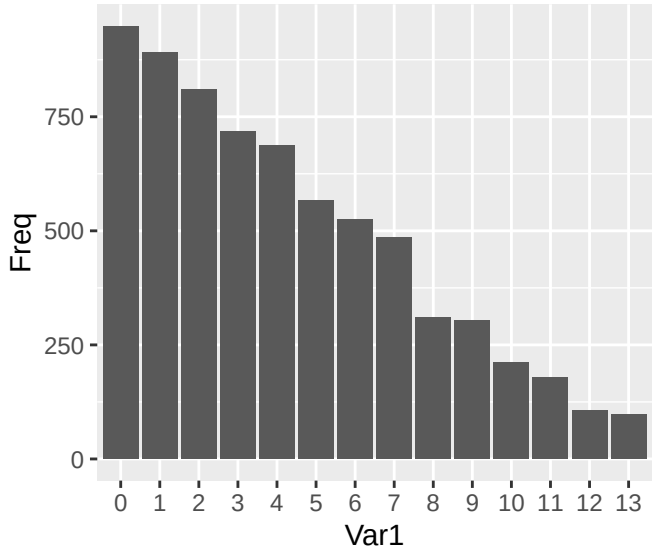
```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13
## 949 891 809 718 687 566 526 486 310 303 212 179 106  97
```

```
cluster.counts.df <- data.frame(cluster.counts)
cluster.counts.df
```

```
##    Var1 Freq
## 1     0  949
## 2     1  891
## 3     2  809
## 4     3  718
## 5     4  687
## 6     5  566
## 7     6  526
## 8     7  486
## 9     8  310
## 10    9  303
```

```
## 11    10  212
## 12    11  179
## 13    12  106
## 14    13   97
```

```
ggplot(cluster.counts.df,aes(x=Var1,y=Freq)) + geom_col()
```



2. Show cells per genotype

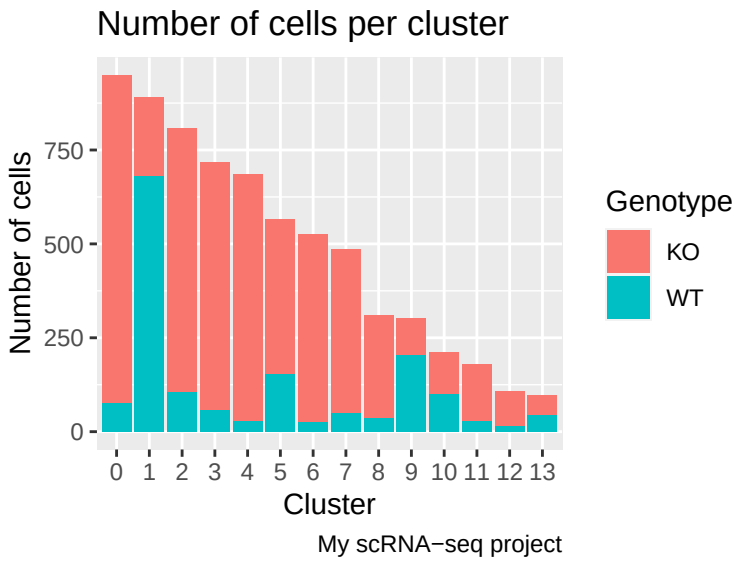
```
ggplot(sc,aes(x=factor(Cluster),fill=Genotype)) + geom_bar()
```

Or if we want the bars side-by-side:

```
ggplot(sc,aes(x=factor(Cluster),fill=Genotype)) + geom_bar(position="dodge")
```

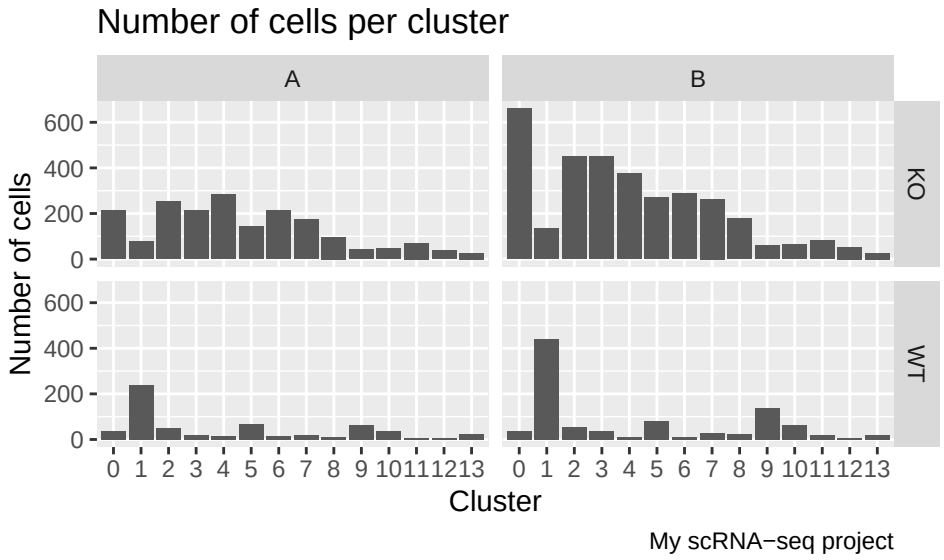
3. Change the labels

```
barTitle <- labs(title="Number of cells per cluster", x="Cluster",
                 y="Number of cells", caption="My scRNA-seq project")

ggplot(sc,aes(x=factor(Cluster),fill=Genotype)) + geom_bar() + barTitle
```



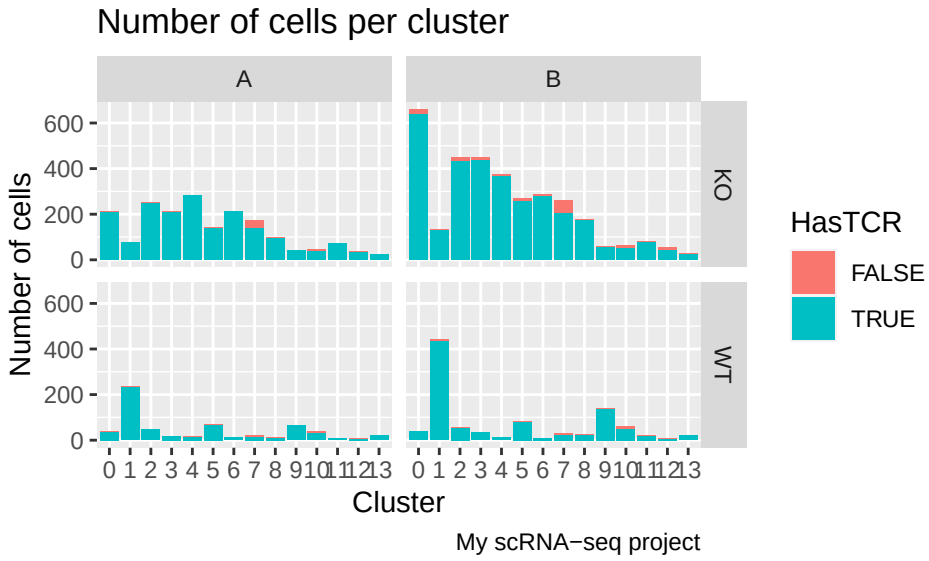
4. Facet the plot by batch and genotype

```
ggplot(sc,aes(x=factor(Cluster))) + geom_bar() + barTitle +
  facet_grid(Genotype ~ Batch)
```



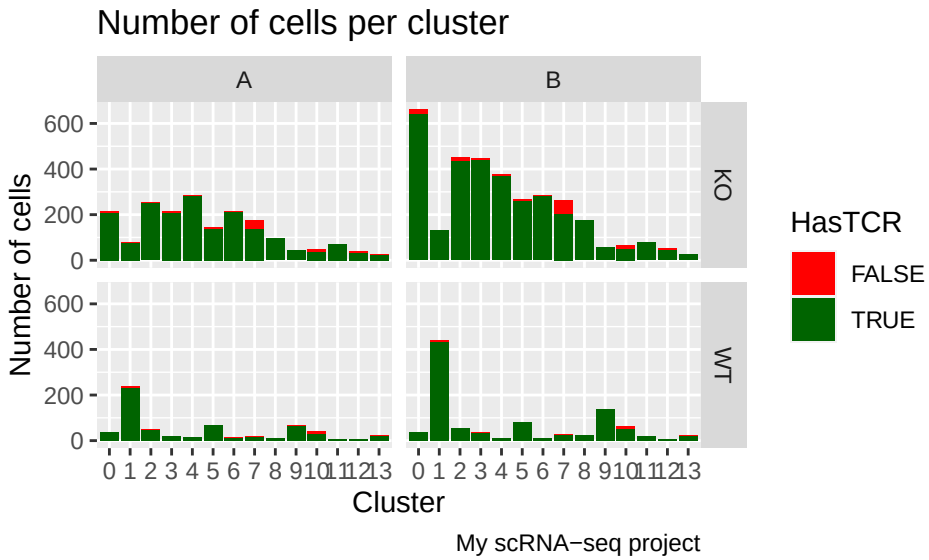
5. Add coloring based on the TCR expression

```
ggplot(sc,aes(x=factor(Cluster),fill=HasTCR)) + geom_bar() + barTitle +
  facet_grid(Genotype ~ Batch)
```



6. Change the color of the fill used to display TCR expression

```
ggplot(sc,aes(x=factor(Cluster),fill=HasTCR)) + geom_bar() + barTitle +
  facet_grid(Genotype ~ Batch) +
  scale_fill_manual(values=c("red", "darkgreen"))
```



2.4 ggplot2 vs built-in R plots (take home exercise)

If you have closed RStudio, make sure to reload the `ggplot2` package.

```
# load the package
library(ggplot2)
```

For this exercise we will recreate the following scatter plot from the mtcars data set using the built-in plot functions in R. The plot will have the following features.

- Scatter plot mtcars with `wt` (x axis) vs. `mpg` (y axis)
- Points will be colored by `cyl` as a factor.
- Point shapes will be set by `am` as a factor.
- Point size will be based on ratio of `hp` / `wt`
- Plot will be split into multi-plot (facets) using `am` and `cyl` as factors.

2.4.1 2.4.1 Create the plot using ggplot2

1. Create baseplot setting color, plot shapes, and point sizes.

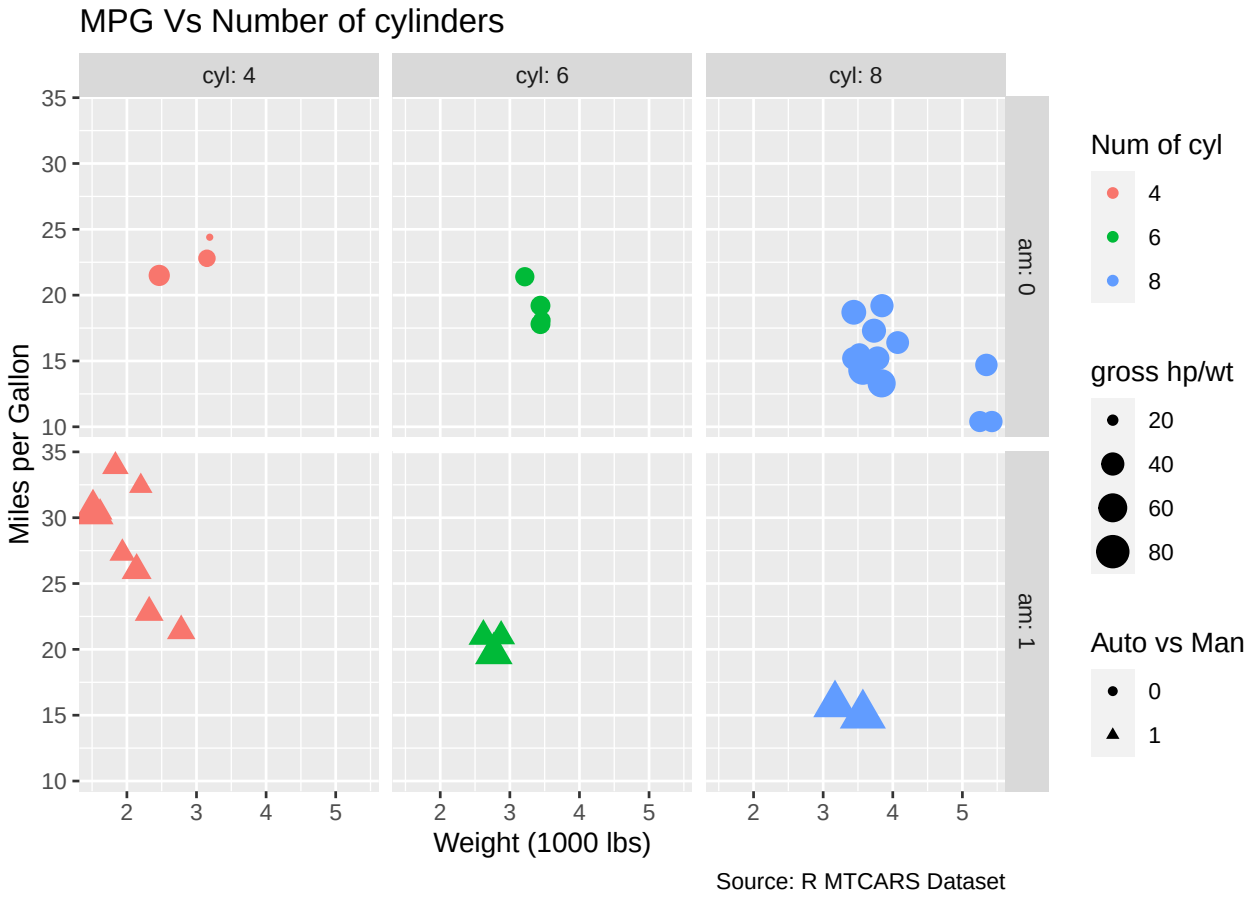
```
basePlot <- ggplot(mtcars, aes(wt, mpg, color=factor(cyl), shape=factor(am),
    size=(hp/wt)))
```

2. Update titles and labels.

```
basePlotTitle <- labs(title="MPG Vs Number of cylinders", y="Miles per Gallon",
                      x="Weight (1000 lbs)", caption="Source: R MTCARS Dataset",
                      shape="Auto vs Man", color="Num of cyl", size="gross hp/wt")
```

3. Generate plot with facet.

```
basePlot + geom_point() + facet_grid(am ~ cyl, labeller = label_both) + basePlotTitle
```



Source: R MTCARS Dataset

2.4.2 2.4.2 Create the plot using built-in R plot functions

1. Define colors for each row in `mtcars`.

- First will create a factor using values of column `cyl`.
- Will changes the levels of the factor to be colors.

```
point_colors <- factor(mtcars$cyl)
colors_legend <- levels(point_colors)
levels(point_colors) <- rainbow(length(levels(point_colors)))
```

2. Define point shapes for each row in `mtcars`

- First will create a factor using values of column `am`.
- Will changes the levels of the factor to numbers for point symbols. Symbols 16 - 18 are solid shapes, so we will put those first.

```
point_shapes <- factor(mtcars$am)
shapes_legend <- levels(point_shapes)
levels(point_shapes) <- c(16:18,0:17)[1:length(levels(point_shapes))]
shapes_legend_levels <- as.numeric(levels(point_shapes))
point_shapes <- as.numeric(as.character(point_shapes))
```

3. Define point sizes for each row in `mtcars`

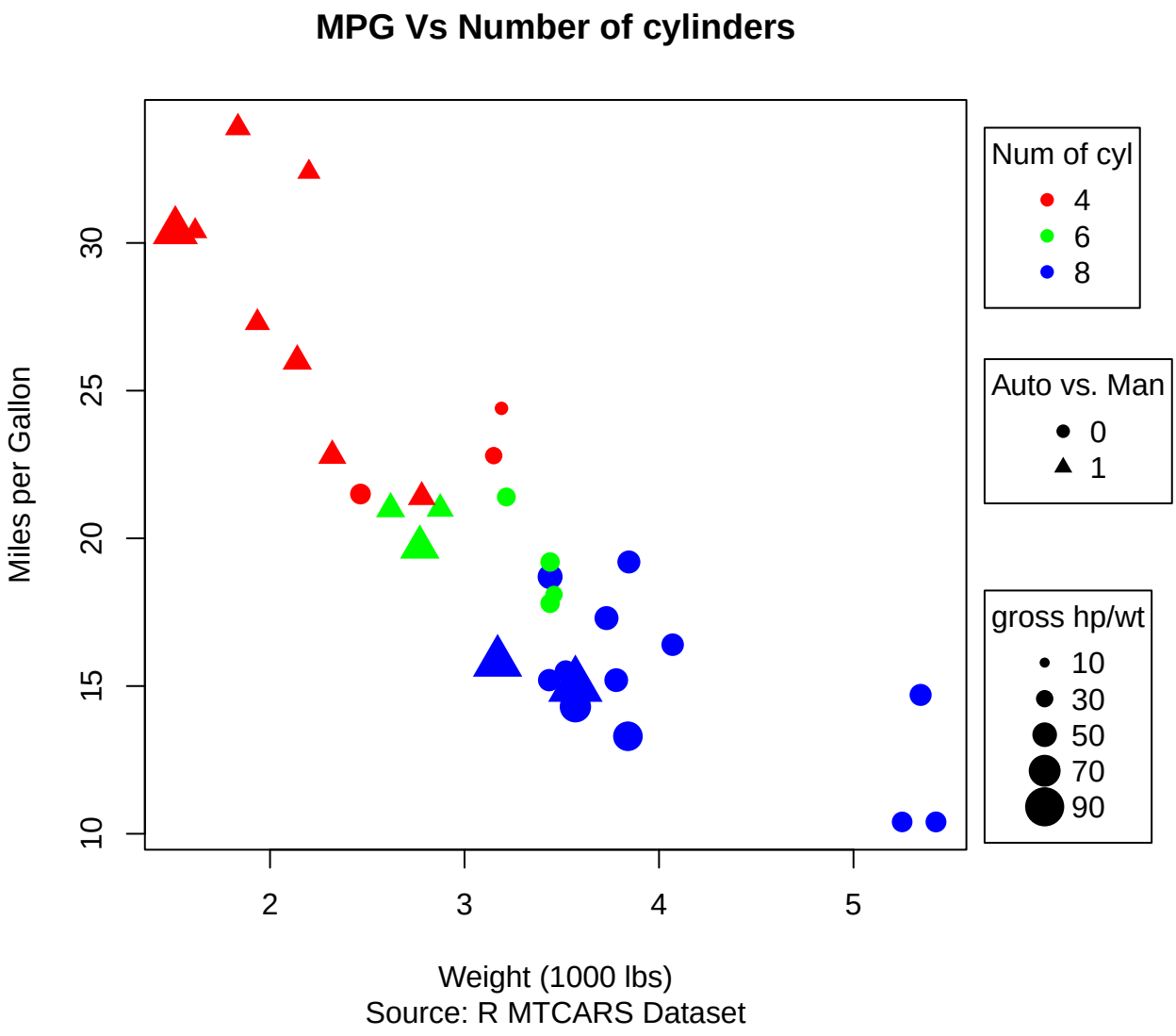
- Will create a vector using computed value of `hp` / `wt`
- Determine range of values in vector
- Then recompute point sizes

```
point_sizes <- mtcars$hp / mtcars$wt
size_range <- range(point_sizes) # Will return minimum and maximum value of vector
size_conv <- (size_range[2] - size_range[1] ) / 2 # Will range sizes over 1-3 (2 units)
point_sizes <- ( ( point_sizes - size_range[1] ) / size_conv ) + 1

# Set the legend breaks approximately every 0.5 point change, rounded to 10s

size_legend <- seq(floor(size_range[1] / 10) * 10, ceiling(size_range[2] / 10) * 10,
                   by=ceiling(size_conv / 20 ) * 10)
size_legend_cex <- ( ( size_legend - size_range[1]) / size_conv ) + 1
```

4. Create the basic plot

- Plot `wt` as x-axis and `mpg` as y-axis
- `type="p"` indicates ploting points
- Point colors are set using `col` parameter
- Point shapes are set using `pch` parameter
- Point sizes are set using `cex` parameter
- Set the title (`main`), caption (`sub`), axis labels (`xlab` and `ylab`)
- Add the appropriate legends

```
# Set the margins to allow room for the legend on the right hand side
par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)

# Create the basic plot
plot(mtcars$wt, mtcars$mpg, type="p",
     col=as.character(point_colors), pch=point_shapes, cex=point_sizes,
     main="MPG Vs Number of cylinders",
     xlab="Weight (1000 lbs)",
     ylab="Miles per Gallon",
     sub="Source: R MTCARS Dataset")

# Set the spacing for the legends. There are three so, space each one a third the way
# on the y axis
legend_spacing <- ( max(mtcars$mpg) - min(mtcars$mpg) ) / 3

# Add the legend for colors
legend(max(mtcars$wt) + 0.25, ( 3 * legend_spacing ) + min(mtcars$mpg),
       legend = colors_legend, col=levels(point_colors),
       pch=16, title="Num of cyl")

# Add the legend for shapes
legend(max(mtcars$wt) + 0.25, (2 * legend_spacing ) + min(mtcars$mpg),
       legend = shapes_legend,
       pch=shapes_legend_levels,
       title="Auto vs. Man")

# Add the legend for sizes
legend(max(mtcars$wt) + 0.25, legend_spacing + min(mtcars$mpg),
       legend = size_legend,
       pch=16, pt.cex=size_legend_cex,
       title="gross hp/wt")
```


MPG Vs Number of cylinders
Miles per Gallon
10
15
20
25
30
2
3
4
5
Weight (1000 lbs)
Source: R MTCARS Dataset
Num of cyl
4
6
8
Auto vs. Man
0
1
gross hp/wt
10
30
50
70
90

5. Facet the plot

The `layout` function allows one to create separate plot areas. We will use this function to create a plot area for each sub-plot (facet).

```
# Get the number of levels for each factor
am_levels <- unique(sort(mtcars$am))
cyl_levels <- unique(sort(mtcars$cyl))

# Create the layout matrix
layout_mat <- matrix(seq(1, length(am_levels) * length(cyl_levels)),
                         nrow=length(am_levels),
                         ncol=length(cyl_levels), byrow = T)

# Look at the layout matrix
layout_mat
```

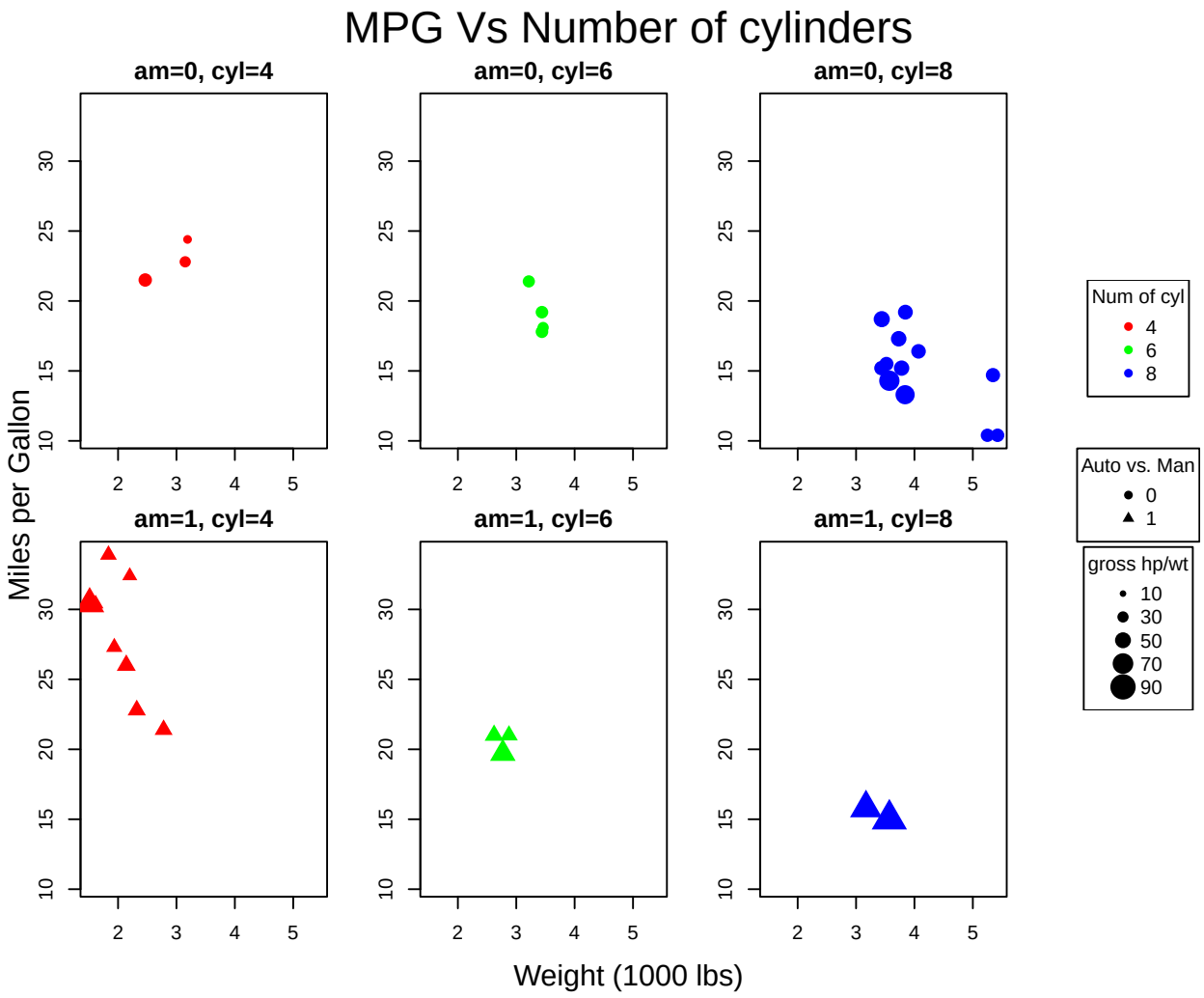
```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
# Add a column for the legends
layout_mat <- cbind(layout_mat,
                    rep((length(cyl_levels) * length(am_levels) + 1), nrow(layout_mat)))

# Look at the final layout matrix
layout_mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    7
## [2,]    4    5    6    7
```

```
# Set outer margins for the title, subtitle (caption) and axis labels
par(oma = c(4, 2, 3, 0))

# Setup the layout using the layout matrix
# Set the widths so that each plot is 1 share and the legend is a half (0.5 share)
layout(mat=layout_mat, widths=c(rep(1, length(cyl_levels)), 0.5))

# Create a data frame of the plot styles. Will use this when subsetting for each subplot
plot_styles <- data.frame(col=as.character(point_colors),
                          pch=point_shapes,
                          cex=point_sizes, stringsAsFactors = F)

# Loop over all values of am and cyl to create each plot.
for ( am_value in am_levels ) {
  for ( cyl_value in cyl_levels ) {
    # Set the margins of the subplot
    par(mar = c(2,2,2,2))
    # Create an empty plot with the same x and y scale.
    plot(mtcars$wt, mtcars$mpg, type="n",
         main=sprintf("am=%d, cyl=%d", am_value, cyl_value))
    # Subset the data (mtcars) and plot styles.
    mtcars_subset <- mtcars[mtcars$am == am_value & mtcars$cyl == cyl_value, ]
    plot_styles_subset <- plot_styles[mtcars$am == am_value & mtcars$cyl == cyl_value, ]
    # Add the data for this subplot
```

```
    points(mtcars_subset$wt, mtcars_subset$mpg,
     col=plot_styles_subset$col,
     pch=plot_styles_subset$pch,
     cex=plot_styles_subset$cex)
  }
}

# Create a new plot for the legends
par(mar = c(10,1,10,1))
plot(1, type="n", axes=F, xlab="", ylab="")

# Put the colors legend at the top of the area
legend(x="top",
       legend = colors_legend, col=levels(point_colors),
       pch=16, title="Num of cyl")

# Put the shapes legend in the center of the area
legend(x="center",
       legend = shapes_legend,
       pch=shapes_legend_levels,
       title="Auto vs. Man")

# Put the size legend are the bottom of the area
legend(x="bottom",
       legend = size_legend,
       pch=16, pt.cex=size_legend_cex,
       title="gross hp/wt")

# Add the title and subtitle (caption)
mtext("MPG Vs Number of cylinders", outer = TRUE, cex = 1.5)
mtext("Source: R MTCARS Dataset", outer = TRUE, side=1, cex=0.75, line=3)

# Add the axis labels
mtext("Weight (1000 lbs)", outer=TRUE, cex=1, line=1, side=1)
mtext("Miles per Gallon", outer=TRUE, cex=1, side=2)
```



Source: R MTCARS Dataset

3 PART 3

3.1 Descriptive statistics

NOTE: If you have not previously installed `doBy` then install from CRAN.

```
install.packages("doBy")
```

Once the `doBy` package is installed make sure to load for this exercise

```
# load the package
library(doBy)
```

```
bw_data <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/birth_weight.txt",
   header=T)
head(bw_data)
```

```
##   bwt gestation parity age height weight smoke
## 1 120       284      0  27     62    100     0
## 2 113       282      0  33     64    135     0
## 3 128       279      0  28     64    115     1
## 4 108       282      0  23     67    125     1
## 5 136       286      0  25     62     93     0
## 6 138       244      0  33     62    178     0
```

Use `apply` to summarize descriptive statistics

```
# obtain mean for all variables using `apply` function
# the 2nd argument "2" means column wise.
apply(bw_data, 2, mean)
```

```
##         bwt   gestation      parity         age      height      weight
## 119.4625213 279.1013629   0.2623509  27.2282794  64.0494037 128.4787053
##       smoke
##   0.3909710
```

```
# obtain median for all variables using `apply` function
apply(bw_data, 2, median)
```

```
##       bwt gestation    parity       age    height    weight     smoke
##       120       280         0        26        64       125         0
```

```
# obtain standard deviation for all variables using `apply` function
apply(bw_data, 2, sd)
```

```
##        bwt  gestation     parity        age     height     weight      smoke
## 18.3286714 16.0103051  0.4400999  5.8178387  2.5261015 20.7342822  0.4881759
```

Use `summaryBy` to summarize groupwise statistics

```
# use "summaryBy" to obtain groupwise statistics
summaryBy(bwt~smoke, data=bw_data, FUN=c(mean, sd, min, median, max))
```

```
  smoke bwt.mean   bwt.sd bwt.min bwt.median bwt.max
1     0 123.0853 17.42370      55        123     176
2     1 113.8192 18.29501      58        115     163
```

```
summaryBy(bwt~parity, data=bw_data, FUN=c(mean, sd, min, median, max))
```

```
  parity bwt.mean   bwt.sd bwt.min bwt.median bwt.max
1      0 119.9423 18.66204      55        120     174
2      1 118.1136 17.31515      63        118     176
```

Use `table` to create a contingency table

```
# build a contingency table of the counts at each combination of factor levels.
# parity: 0 - child first born, 1 - otherwise
# smoke: 0 - mother is not a smoker, 1 - smoker
table(bw_data$parity, bw_data$smoke, dnn=c("parity", "smoke"))
```

```
##       smoke
## parity   0   1
##      0 525 341
##      1 190 118
```

3.2 Check normality and ANOVA

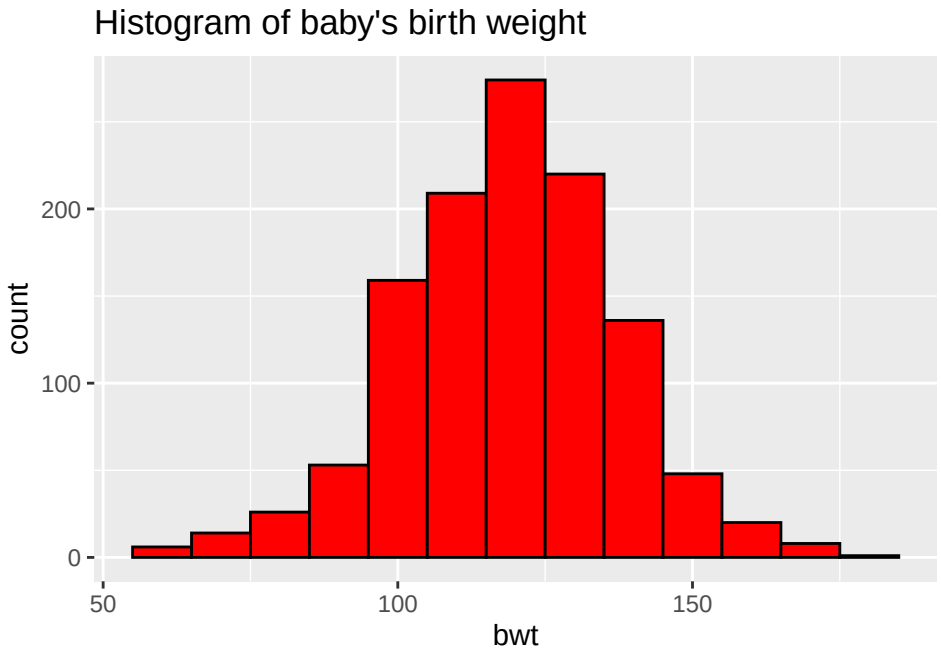
If you have closed RStudio, make sure to reload the `ggplot2` package.

```
# load the package
library(ggplot2)
```

```
bw_data <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/birth_weight.txt", header=T)
```

1. Use `ggplot` to create a histogram of baby's birth weight

```
# check if it is a bell shape in the histogram of baby's birth weight
ggplot(bw_data, aes(x=bwt)) +
        geom_histogram(binwidth=10, fill="red", color="black") +
        labs(title="Histogram of baby's birth weight")
```



2. Use `shapiro.test` to check normality of baby's birth weight

Caution: a large sample size will usually result in a significant p value in Shapiro test, because any tiny deviation from normality will be detected. We should also look at how far W is from 1 to see how big the deviation is. For reasonably small differences, the normality assumption is probably still OK.

```
# there are 1174 data points
length(bw_data$bwt)
```

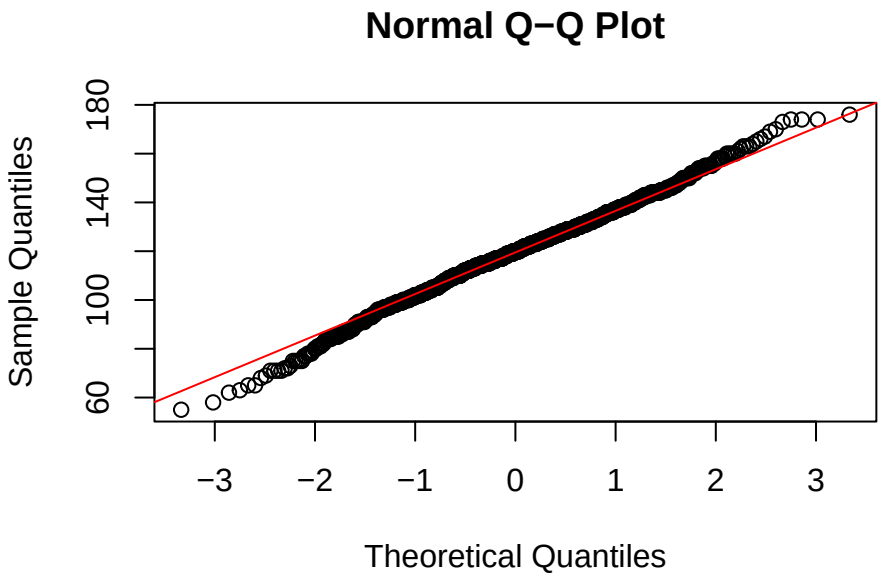
```
## [1] 1174
```

```
shapiro.test(bw_data$bwt)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  bw_data$bwt
## W = 0.99563, p-value = 0.001917
```

The p-value is <0.05, but W is quite close to 1, so we're probably OK. Look at the Q-Q plot also.

```r
# use QQ-normal plot to check the normality.
# if the data is normally distributed, the points in the QQ-normal plot will
# lie on a straight diagonal line.
qqnorm(bw_data$bwt)
# qqline shows a line for a "theoretical" normal distribution
qqline(bw_data$bwt, col="red")
```

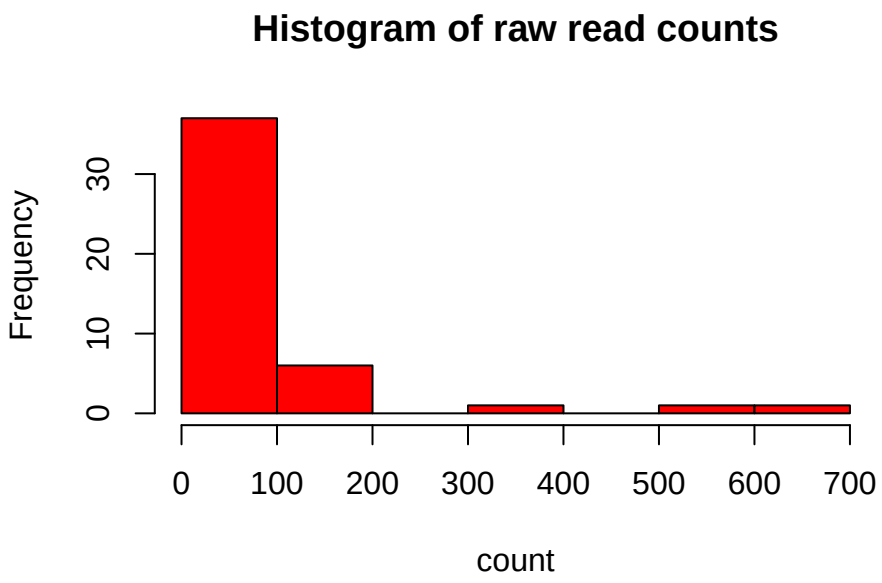


3. Use `shapiro.test` to check normality of a raw count of a gene from RNA-seq

```r
# get raw read count data
raw.count <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/raw.count.txt",
            header=T, row.names=1)
# convert to the gene's raw read count to a numeric vector
count <- as.numeric(raw.count[1, ])
# there are 46 data points
length(count)
```

```
## [1] 46
```

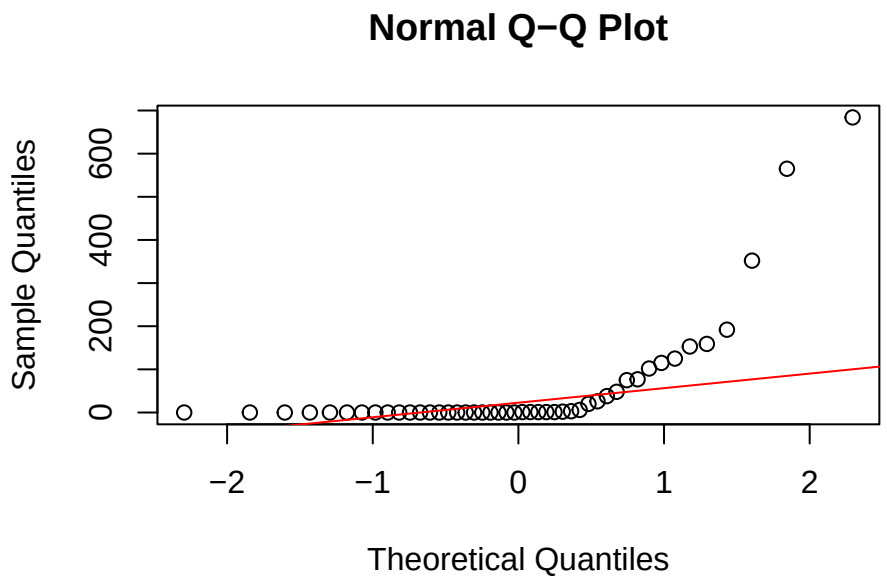
```r
# check if it is a bell shape in the histogram of raw read counts
hist(count, main="Histogram of raw read counts", col="red")
```




```
shapiro.test(count)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  count
## W = 0.49027, p-value = 2.014e-11
```

```
qqnorm(count)
# qqline adds a line for a "theoretical" normal distribution
qqline(count, col="red")
```

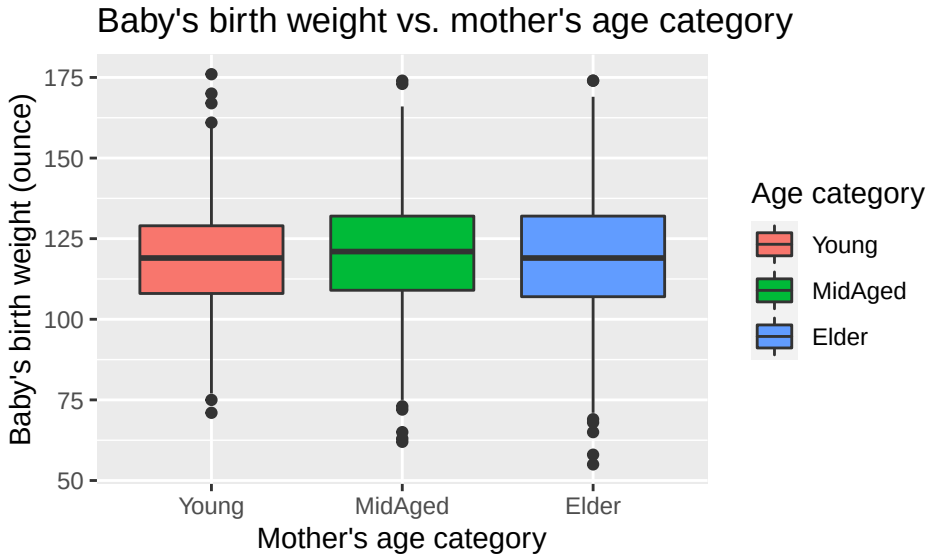


4. One way ANOVA to test baby's birth weight vs mother's age category

```
# create a new column for mother's age category
bw_data$agecat <- cut(bw_data$age,
    breaks=c(0,23,30,Inf), labels=c("Young", "MidAged", "Elder"))
head(bw_data)
```

```
##   bwt gestation parity age height weight smoke  agecat
## 1 120       284      0  27     62    100     0 MidAged
## 2 113       282      0  33     64    135     0   Elder
## 3 128       279      0  28     64    115     1 MidAged
## 4 108       282      0  23     67    125     1   Young
## 5 136       286      0  25     62     93     0 MidAged
## 6 138       244      0  33     62    178     0   Elder
```

```
# create a boxplot: baby's birth weight ~ mother's three age categories.
boxTitle <- labs(title="Baby's birth weight vs. mother's age category",
    y="Baby's birth weight (ounce)", x="Mother's age category", fill="Age category")
ggplot(bw_data, aes(x=agecat, y=bwt)) + geom_boxplot(aes(fill=agecat)) + boxTitle
```




```
# One way ANOVA to test the baby's birth weight among mother's three age categories.
summary(aov(bwt ~ agecat, data=bw_data))
```

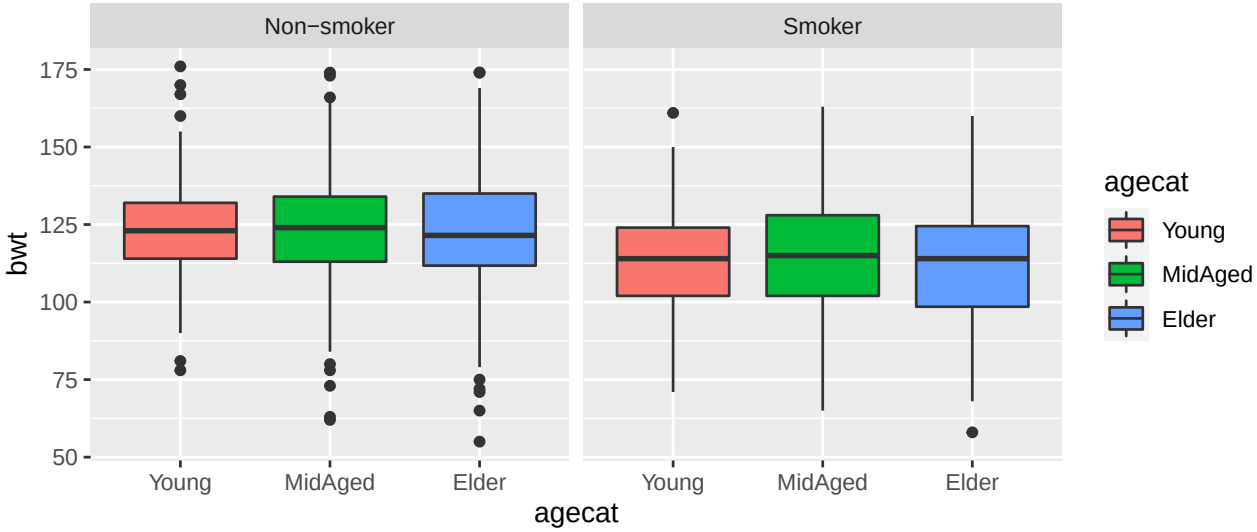
```
##              Df Sum Sq Mean Sq F value Pr(>F)
## agecat        2    728   364.1   1.084  0.339
## Residuals  1171 393330   335.9
```

```
# Redirect this to a variable so we can save the statistics.
anova.stats <- summary(aov(bwt ~ agecat, data=bw_data))
anova.pvalues <- anova.stats[[1]][["Pr(>F)"]]
anova.pvalues
```

```
## [1] 0.3386306        NA
```

5. Two way ANOVA

```
# label smoking status: 0 - non-smoker; 1 - smoker
bw_data$smoke <- factor(bw_data$smoke, labels=c("Non-smoker", "Smoker"))
# create a boxplot using facet grid of smoke (0 or 1) and age category
ggplot(bw_data, aes(x=agecat, y=bwt, group=agecat)) +
        geom_boxplot(aes(fill=agecat)) + facet_grid( ~ smoke)
```




```
# Two way ANOVA: agecat and smoke
summary(aov(bwt ~ agecat + smoke, data=bw_data))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## agecat         2    728     364   1.153  0.316
## smoke          1  23894   23894  75.671 <2e-16 ***
## Residuals   1170 369436     316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6. Two way ANOVA with an interaction term

```
# Two way ANOVA with an interaction term
summary(aov(bwt ~ agecat * smoke, data=bw_data))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## agecat          2    728     364   1.153  0.316
## smoke           1  23894   23894  75.674 <2e-16 ***
## agecat:smoke    2    642     321   1.017  0.362
## Residuals    1168 368794     316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# save as in a variable
twoway.stats <- summary(aov(bwt ~ agecat * smoke, data=bw_data))
# see p-values
twoway.stats[[1]][["Pr(>F)"]]
```

```
## [1] 3.160488e-01 1.118213e-17 3.618961e-01           NA
```

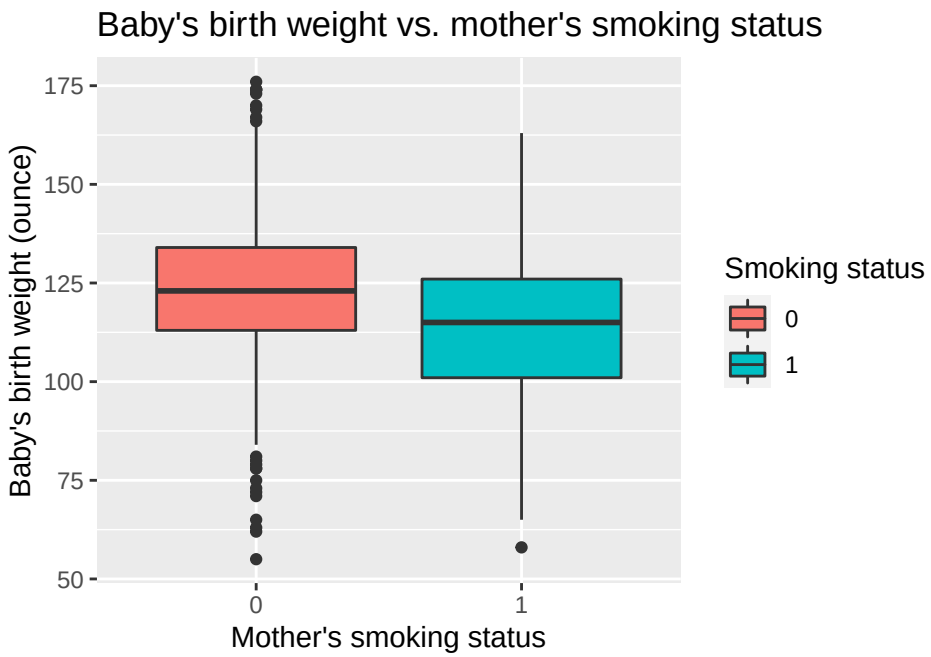
3.3 Wilcoxon rank sum test and Kruskal-Wallis test

If you have closed RStudio, make sure to execute the following.

```
# load the package
library(ggplot2)
bw_data <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/birth_weight.txt", header=T)
# create a new column for mother's age category
bw_data$agecat[bw_data$age > 30] <- "Elder"
bw_data$agecat[bw_data$age > 23 & bw_data$age <= 30] <- "MidAged"
bw_data$agecat[bw_data$age <= 23] <- "Young"
# change agecat to a factor
bw_data$agecat <- factor(bw_data$agecat, levels=c("Young", "MidAged", "Elder"))
```

1. Wilcoxon rank sum test

```
# create a boxplot for baby's birthweight vs. mother's smoking status
boxTitle <- labs(title="Baby's birth weight vs. mother's smoking status",
    y="Baby's birth weight (ounce)", x="Mother's smoking status", fill="Smoking status")
ggplot(bw_data, aes(x=factor(smoke), y=bwt) )  +
        geom_boxplot( aes(fill = factor(smoke))) + boxTitle
```




```
# Use non-parametric test to test baby's birthweight vs. mother's smoking status
wilcox.test(bwt ~ smoke, data=bw_data)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  bwt by smoke
## W = 212970, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
# save as a variable
wilcox.stats <- wilcox.test(bwt ~ smoke, data=bw_data)

# p-value
```

```
wilcox.stats$p.value
```

```
## [1] 6.485236e-18
```

2. Kruskal-Wallis test

```
head(bw_data)
```

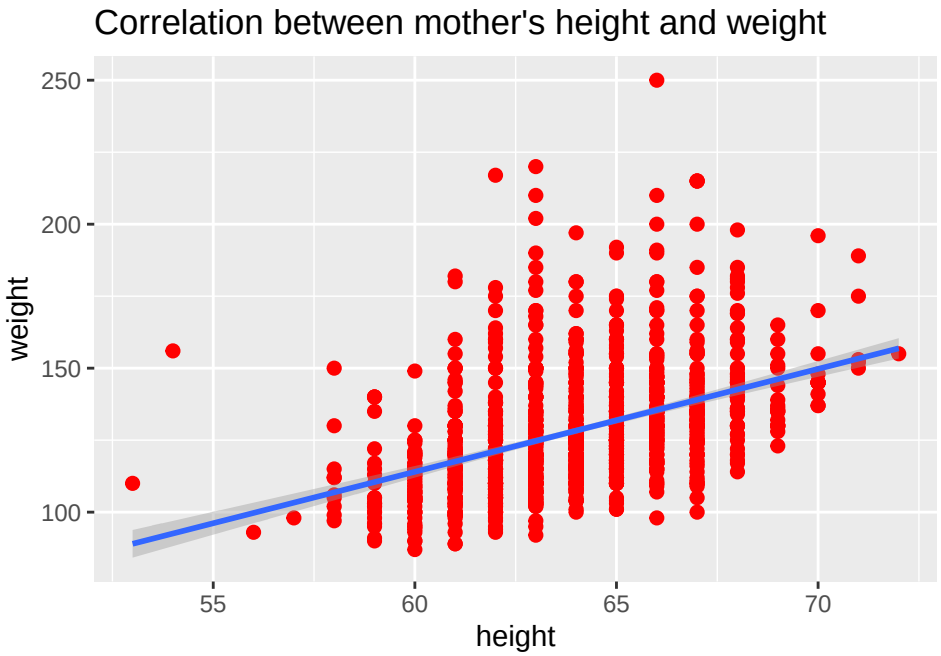
```
##   bwt gestation parity age height weight smoke  agecat
## 1 120       284      0  27     62    100     0 MidAged
## 2 113       282      0  33     64    135     0   Elder
## 3 128       279      0  28     64    115     1 MidAged
## 4 108       282      0  23     67    125     1   Young
## 5 136       286      0  25     62     93     0 MidAged
## 6 138       244      0  33     62    178     0   Elder
```

```
# Use non-parametric test to test baby's birthweight vs. mother's age category
kruskal.test(bwt ~ agecat, data=bw_data)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  bwt by agecat
## Kruskal-Wallis chi-squared = 2.6045, df = 2, p-value = 0.2719
```

3.4 Correlation test

```
# create a scatter plot to see relationship between mother's height and weight
ggplot(bw_data, aes(x=height, y=weight)) +
        geom_point(size=2, color="red") +
        geom_smooth(formula=y~x,method = "lm", se = T) +
        labs(title="Correlation between mother's height and weight",
             x="height", y="weight")
```



geom_smooth() adds a smoothed conditional means to the plot

- formula - Formula to use in the smoothing function
- method - Smoothing method (e.g., lm, glm, gam, loess)
- se - Display confidence interval around smooth.

1. Pearson correlation

```
# Pearson correlation
cor.test(bw_data$height, bw_data$weight)
```

```
##
##  Pearson's product-moment correlation
##
## data:  bw_data$height and bw_data$weight
## t = 16.552, df = 1172, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3877305 0.4805332
## sample estimates:
##       cor
## 0.4352874
```

```
# save as a variable
pearson.stats <- cor.test(bw_data$height, bw_data$weight)
# p-value
pearson.stats$p.value
```

```
## [1] 1.837015e-55
```

```
# correlation coefficient
pearson.stats$estimate
```

```
##       cor
## 0.4352874
```

2. Spearman correlation

```
# Spearman correlation
cor.test(bw_data$height, bw_data$weight, method="spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  bw_data$height and bw_data$weight
## S = 134713533, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.5004735
```

3. Kendall correlation

```
# Kendall correlation
cor.test(bw_data$height, bw_data$weight, method="kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  bw_data$height and bw_data$weight
## z = 18.02, p-value < 2.2e-16
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.3743995
```

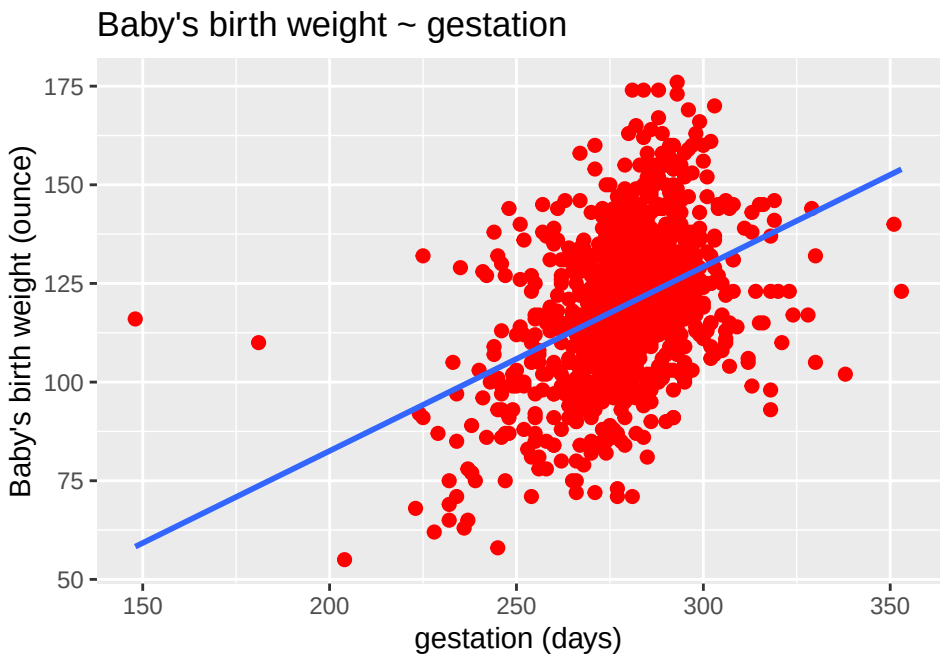
3.5 Linear regression

1. Simple linear model

```
# simple linear model
model <- lm(bwt ~ gestation, data=bw_data) # fit the regression model
# display model results
summary(model)
```

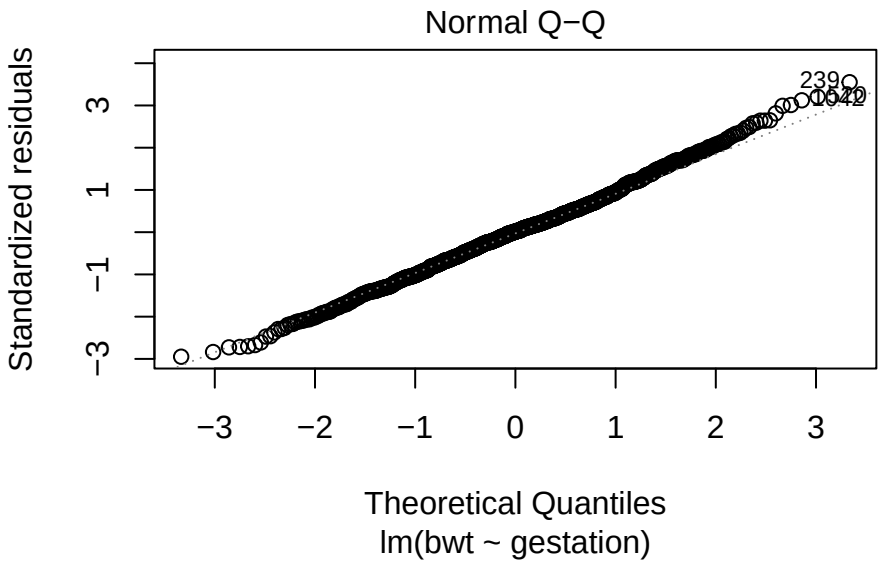
```
##
## Call:
## lm(formula = bwt ~ gestation, data = bw_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.348 -11.065   0.218  10.101  57.704
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.75414    8.53693   -1.26    0.208
## gestation     0.46656    0.03054   15.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.74 on 1172 degrees of freedom
## Multiple R-squared:  0.1661, Adjusted R-squared:  0.1654
## F-statistic: 233.4 on 1 and 1172 DF,  p-value: < 2.2e-16
```

```
# create a scatter plot for baby's bwt vs gestation
ggplot(bw_data, aes(x=gestation, y=bwt)) +
        geom_point(size=2, color="red") +
        geom_smooth(formula=y~x,method = "lm", se = F) +
        labs(title="Baby's birth weight ~ gestation", x="gestation (days)",
             y="Baby's birth weight (ounce)")
```



2. Normal Q-Q plot of residuals

```
# There will be four plots from plot(model). The 2nd one is Q-Q plot.
# The Q-Q plot show that residuals are normally distributed,
# which means that the data meets normality assumptions of linear regression.
plot(model, 2)
```

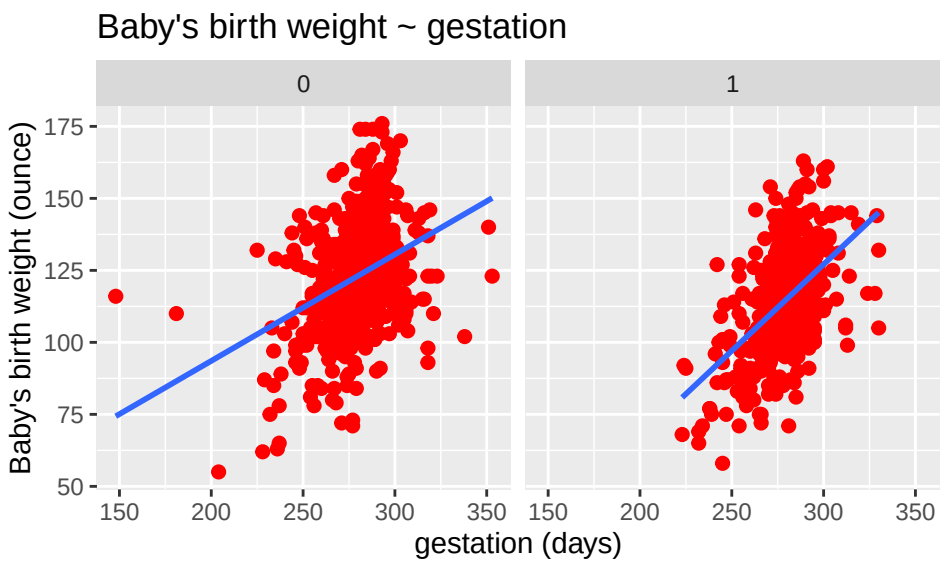


3. General linear model for multiple factors

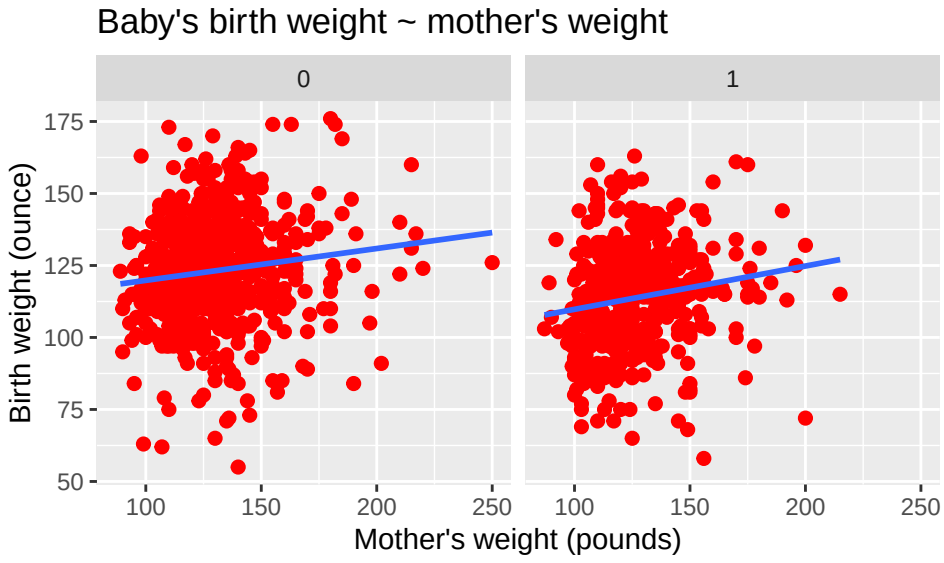
```
# general linear model for multiple factors
model <- lm(bwt ~ gestation + weight + factor(smoke), data=bw_data)
# display model results
summary(model)
```

```
##
## Call:
## lm(formula = bwt ~ gestation + weight + factor(smoke), data = bw_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.920 -10.759  -0.279   9.743  51.354
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -17.62648    8.69128  -2.028   0.0428 *
## gestation        0.44809    0.02936  15.261  < 2e-16 ***
## weight           0.11818    0.02267   5.213  2.2e-07 ***
## factor(smoke)1  -8.07789    0.96444  -8.376  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.07 on 1170 degrees of freedom
## Multiple R-squared:  0.2335, Adjusted R-squared:  0.2315
## F-statistic: 118.8 on 3 and 1170 DF,  p-value: < 2.2e-16
```

```
# create scatter plot for baby's bwt vs gestation, facet by smoke
gg1 <- ggplot(bw_data, aes(x=gestation, y=bwt)) +
        geom_point(size=2, color="red") +
        geom_smooth(formula=y~x,method = "lm", se = F) +
        labs(title="Baby's birth weight ~ gestation", x="gestation (days)",
        y="Baby's birth weight (ounce)")
gg1 + facet_grid(~smoke)
```




```
# create scatter plot for baby's bwt vs weight, facet by smoke
gg2 <- ggplot(bw_data, aes(x=weight, y=bwt)) +
        geom_point(size=2, color="red") +
        geom_smooth(formula=y~x,method = "lm", se = F) +
        labs(title="Baby's birth weight ~ mother's weight", x="Mother's weight (pounds)",
        y="Birth weight (ounce)")
gg2 + facet_grid(~smoke)
```



3.6 Chi-squared test

Is there a significant association between women's age at first birth and breast cancer status?

	<20	20-24	25-29	30-34	>34
Cancer	320	1206	1011	463	220
No cancer	1422	4432	2893	1092	406

```
# read the data into a data frame
age_cancer <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/age_cancer.txt",
        header=T, sep="\t", row.names=1, check.names=F)
age_cancer
```

```
##            <20 20-24 25-29 30-34 >=35
## Cancer     320  1206  1011   463  220
## No cancer 1422  4432  2893  1092  406
```

```
chisq.test(age_cancer)
```

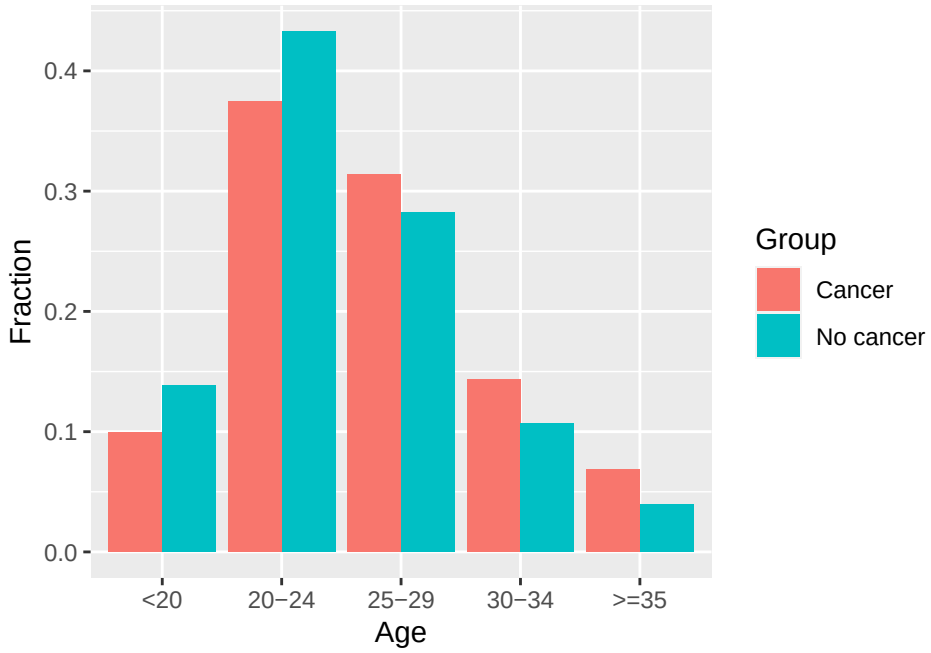
```
##
##  Pearson's Chi-squared test
##
## data:  age_cancer
## X-squared = 130.34, df = 4, p-value < 2.2e-16
```

Make a bar plot of the fraction of cancer vs non-cancer patients in each age category.

```
# normalize counts to percent (across each row)
age_cancer_percent <- age_cancer / rowSums(age_cancer)
# add a column for the groups
age_cancer_percent$Group = rownames(age_cancer_percent)
# melt into long format
library(reshape2)
age_cancer_melt <- melt(age_cancer_percent)
```

```
## Using Group as id variables
```

```
colnames(age_cancer_melt)[2:3] <- c("Age","Fraction")
# make the plot
ggplot(age_cancer_melt,aes(x=Age, y=Fraction, fill=Group)) +
   geom_col(position='dodge')
```



A bit more programming: write a loop to run a post-hoc with Fisher's Exact test for each age group. Note: you do not need to type the comments in the loop, but they are there to explain what each command is doing.

```
# store number of columns
cols <- ncol(age_cancer)
# start an empty data frame
age_stats <- data.frame()
# run a test for each column (age group) one at a time
for(i in 1:cols){
  # get the counts for this age group
  age.counts <- age_cancer[,i]
  # get the counts for all other age groups
  age.other <- rowSums(age_cancer[,1:cols != i])
  # we're specifically seeing if THIS age group has a different distribution
  # of cancer vs non-cancer patients, using fisher's exact test
  age.table <- cbind(age.counts, age.other)
  fet <- fisher.test(age.table)
  # combine the estimate (odds ratio) and p-value) into the data frame
  # and use the age group as the row name
  age_stats <- rbind(age_stats, c(fet$estimate, fet$p.value) )
  rownames(age_stats)[nrow(age_stats)] <- colnames(age_cancer)[i]
}
# set the column names, and add log2-scaled odds ratio and FDR corrected p-value
colnames(age_stats) <- c("OddsRatio","P.Value")
age_stats$Log2OddsRatio = log2(age_stats$OddsRatio)
age_stats$Q.Value = p.adjust(age_stats$P.Value,method="fdr")
age_stats
```

```
##       OddsRatio      P.Value Log2OddsRatio      Q.Value
## <20   0.6846664 2.858742e-09    -0.5465269 7.146854e-09
## 20-24 0.7854116 5.275017e-09    -0.3484792 8.791695e-09
## 25-29 1.1630615 6.062557e-04     0.2179274 6.062557e-04
## 30-34 1.4075803 1.742257e-08     0.4932172 2.177821e-08
## >=35  1.7770394 9.774348e-11     0.8294756 4.887174e-10
```

The young age groups have odds ratios <1: lower fraction in cancer groups, which we can also see in the bar plot. The biggest deviation, based on the log2 odds ratio, is for the oldest group. All of the groups have statistically significant differences.

Exercise: do the same thing with an apply statement. We use sapply to loop over all elements of a vector, in this case the names of the columns.

```
age_stats2 <- sapply(colnames(age_cancer),function(x){
  age.counts <- age_cancer[[x]]
  age.other <- rowSums(age_cancer[,colnames(age_cancer) != x])
  age.table <- cbind(age.counts, age.other)
  fet <- fisher.test(age.table)
  return(c(fet$estimate, fet$p.value))
})
# the result will have the groups in the columns
# transpose and save as a data frame
age_stats2 <- as.data.frame(t(age_stats2))
# add in the other columns as before
colnames(age_stats2) <- c("OddsRatio","P.Value")
age_stats2$Log2OddsRatio = log2(age_stats2$OddsRatio)
age_stats2$Q.Value = p.adjust(age_stats2$P.Value,method="fdr")
age_stats2
```

```
##       OddsRatio      P.Value Log2OddsRatio      Q.Value
## <20   0.6846664 2.858742e-09    -0.5465269 7.146854e-09
## 20-24 0.7854116 5.275017e-09    -0.3484792 8.791695e-09
## 25-29 1.1630615 6.062557e-04     0.2179274 6.062557e-04
## 30-34 1.4075803 1.742257e-08     0.4932172 2.177821e-08
## >=35  1.7770394 9.774348e-11     0.8294756 4.887174e-10
```

3.7 Read/write data from/to Excel spreadsheets

NOTE: If you have not previously installed `readxl` and `openxlsx` then install from CRAN.

```
install.packages(c("readxl", "openxlsx"))
```

3.7.1 Read Excel spreadsheets using `readxl`

1. Download the Excel spreadsheet from

`https://uic-ric.github.io/workshop-data/advanced_R/taxa_relative.xlsx`

and save to your working directory.

2. Read the first sheet (tab)

```
library(readxl)
sheet_1 <- read_excel("taxa_relative.xlsx", sheet=1)
head(sheet_1)
```

```
## # A tibble: 6 x 33
##   `#OTU ID`      Sample1_M1_Fec1 Sample2_M2_Fec2 Sample3_M3_Fec2 Sample4_M4_Fec2
##   <chr>                    <dbl>           <dbl>           <dbl>           <dbl>
## 1 Bacteria;Acti~         0             0.0000931        0.000388        0.000325
## 2 Bacteria;Bact~         0.512         0.502            0.499           0.441
## 3 Bacteria;Defe~         0             0                0               0
## 4 Bacteria;Firm~         0.486         0.463            0.476           0.539
## 5 Bacteria;Prot~         0             0.000229         0.000523        0.000476
## 6 Bacteria;Tene~         0.00279       0.0347           0.0243          0.0199
## # ... with 28 more variables: Sample5_M5_Fec1 <dbl>, Sample6_M6_Fec3 <dbl>,
## #   Sample7_M7_Fec2 <dbl>, Sample8_M8_Fec2 <dbl>, Sample9_M9_Fec1 <dbl>,
## #   Sample10_M10_Fec2 <dbl>, Sample11_M12_Fec3 <dbl>, Sample12_M13_Fec1 <dbl>,
## #   Sample13_M14_Fec3 <dbl>, Sample14_M15_Fec1 <dbl>, Sample15_M16_Fec2 <dbl>,
## #   Sample16_M18_Fec3 <dbl>, Sample17_M19_Fec2 <dbl>, Sample18_M20_Fec3 <dbl>,
## #   Sample19_M23_Fec2 <dbl>, Sample20_M24_Fec3 <dbl>, Sample21_M25_Fec2 <dbl>,
## #   Sample22_M26_Fec1 <dbl>, Sample23_M29_Fec3 <dbl>, ...
```

3. Get the sheet (tab) names

```
excel_sheets("taxa_relative.xlsx")
```

```
## [1] "L2" "L3" "L4" "L5" "L6" "L7"
```

4. Load the sheet (tab) named "L6"

```
sheet_L6 <- read_excel("taxa_relative.xlsx", sheet="L6")
head(sheet_L6)
```

```
## # A tibble: 6 x 33
##   `#OTU ID`      Sample1_M1_Fec1 Sample2_M2_Fec2 Sample3_M3_Fec2 Sample4_M4_Fec2
##   <chr>                    <dbl>           <dbl>           <dbl>           <dbl>
## 1 Bacteria;Acti~               0       0               0               0
## 2 Bacteria;Acti~               0       0.0000593       0.000388        0.000175
## 3 Bacteria;Acti~               0       0               0               0
## 4 Bacteria;Acti~               0       0.0000339       0               0
## 5 Bacteria;Acti~               0       0               0               0.000151
## 6 Bacteria;Bact~               0       0               0               0.000405
## # ... with 28 more variables: Sample5_M5_Fec1 <dbl>, Sample6_M6_Fec3 <dbl>,
## #   Sample7_M7_Fec2 <dbl>, Sample8_M8_Fec2 <dbl>, Sample9_M9_Fec1 <dbl>,
## #   Sample10_M10_Fec2 <dbl>, Sample11_M12_Fec3 <dbl>, Sample12_M13_Fec1 <dbl>,
```

```
## #   Sample13_M14_Fec3 <dbl>, Sample14_M15_Fec1 <dbl>, Sample15_M16_Fec2 <dbl>,
## #   Sample16_M18_Fec3 <dbl>, Sample17_M19_Fec2 <dbl>, Sample18_M20_Fec3 <dbl>,
## #   Sample19_M23_Fec2 <dbl>, Sample20_M24_Fec3 <dbl>, Sample21_M25_Fec2 <dbl>,
## #   Sample22_M26_Fec1 <dbl>, Sample23_M29_Fec3 <dbl>, ...
```

3.7.2 Read/write Excel spreadsheets using `openxlsx`

1. Read the first sheet (tab)

```
library(openxlsx)
sheet_1 <- read.xlsx("taxa_relative.xlsx", 1)
sheet_1[1:10, 1:5]
```

```
##                       #OTU.ID Sample1_M1_Fec1 Sample2_M2_Fec2 Sample3_M3_Fec2
## 1     Bacteria;Actinobacteria     0.000000000    9.312406e-05    0.0003883167
## 2      Bacteria;Bacteroidetes     0.511541935    5.021842e-01    0.4989701165
## 3    Bacteria;Deferribacteres     0.000000000    0.000000e+00    0.0000000000
## 4         Bacteria;Firmicutes     0.485664296    4.627673e-01    0.4757892960
## 5     Bacteria;Proteobacteria     0.000000000    2.285772e-04    0.0005233834
## 6        Bacteria;Tenericutes     0.002793769    3.472681e-02    0.0243288874
## NA                       <NA>              NA              NA              NA
## NA.1                     <NA>              NA              NA              NA
## NA.2                     <NA>              NA              NA              NA
## NA.3                     <NA>              NA              NA              NA
##      Sample4_M4_Fec2
## 1       0.0003254898
## 2       0.4405544441
## 3       0.0000000000
## 4       0.5387571052
## 5       0.0004763266
## 6       0.0198866343
## NA                NA
## NA.1              NA
## NA.2              NA
## NA.3              NA
```

2. Load the sheet (tab) named "L6"

```
sheet_L6 <- read.xlsx("taxa_relative.xlsx", sheet = "L6")
sheet_L6[1:10, 1:5]
```

```
##                                                                                #OTU.ID
## 1    Bacteria;Actinobacteria;Actinobacteria;Micrococcales;Cellulomonadaceae;Cellulomonas
## 2  Bacteria;Actinobacteria;Coriobacteriia;Coriobacteriales;Eggerthellaceae;Adlercreutzia
## 3       Bacteria;Actinobacteria;Coriobacteriia;Coriobacteriales;Eggerthellaceae;DNF00809
## 4          Bacteria;Actinobacteria;Coriobacteriia;Coriobacteriales;Eggerthellaceae;Other
## 5    Bacteria;Actinobacteria;Coriobacteriia;Coriobacteriales;Eggerthellaceae;Parvibacter
## 6                Bacteria;Bacteroidetes;Bacteroidia;Bacteroidales;Muribaculaceae;CAG-873
## 7                  Bacteria;Bacteroidetes;Bacteroidia;Bacteroidales;Muribaculaceae;Other
## 8                        Bacteria;Bacteroidetes;Bacteroidia;Bacteroidales;Other;Other
## 9             Bacteria;Bacteroidetes;Bacteroidia;Bacteroidales;Porphyromonadaceae;Other
## 10        Bacteria;Bacteroidetes;Bacteroidia;Bacteroidales;Prevotellaceae;Alloprevotella
##    Sample1_M1_Fec1 Sample2_M2_Fec2 Sample3_M3_Fec2 Sample4_M4_Fec2
## 1        0.0000000    0.000000e+00    0.0000000000    0.0000000000
## 2        0.0000000    5.926076e-05    0.0003883167    0.0001746531
## 3        0.0000000    0.000000e+00    0.0000000000    0.0000000000
```

```
## 4        0.0000000    3.386329e-05    0.0000000000    0.0000000000
## 5        0.0000000    0.000000e+00    0.0000000000    0.0001508367
## 6        0.0000000    0.000000e+00    0.0000000000    0.0004048776
## 7        0.5115419    5.020064e-01    0.4989701165    0.4401495665
## 8        0.0000000    0.000000e+00    0.0000000000    0.0000000000
## 9        0.0000000    8.465823e-05    0.0000000000    0.0000000000
## 10       0.0000000    0.000000e+00    0.0000000000    0.0000000000
```

3. Write a data frame to an Excel spreadsheet

```
# Load the "messy" data table from earlier
patients <- read.table("https://uic-ric.github.io/workshop-data/advanced_R/clinical.data.txt",
            header=T, sep="\t")

# Create a workbook with two empty sheets
wb <- createWorkbook()
addWorksheet(wb, "original")
addWorksheet(wb, "clean")

# Write the "messy" data table to the sheet named original
writeData(wb, sheet="original", x=patients)
```

4. Write the "cleaned" data table to the sheet named clean

```
# Load the "clean" data we saved earlier
clean_clinical_data <- read.delim("clean.clinical.data.txt")

writeData(wb, sheet="clean", x=clean_clinical_data)

# Save the workbook to an XLSX file
saveWorkbook(wb, "clinical_data.xlsx", overwrite=T)
```