Intro to Data Visualization and Statistics in R

Session #3

Genome Institute of Singapore

10th October 2019

RECAP SESSION 2 & ASSIGNMENT

What we learnt last week

1. `melt()`
2. `tabyl()`
3. Tidyverse grammar:
   select, filter, group_by, summarize, mutate, arrange
4. Capstone project

Learning objectives for Session 3

This week we will learn more about working with multiple files.

Objectives:

1. Review of assignment
2. Working with missing values
3. `cbind()`, `rbind()`, `join()`

Exercise 1: Genotype frequency

rs1229984 is a risk variant for various cancers. Freq. of T-allele is 70% and C-allele is 30% in East Asians. If you screened 100 unrelated East Asians, how many would carry the TT, CT and CC genotypes?

Exercise 1: Genotype frequency

rs1229984 is a risk variant for various cancers. Freq. of T-allele is 70% and C-allele is 30% in East Asians. If you screened 100 unrelated East Asians, how many would carry the TT, CT and CC genotypes?

From mother	From Father	Genotype	Probability
T	T	TT	0.49 (= 0.7 x 0.7)
C	T	CT	0.21 (= 0.3 x 0.7)
T	C	CT	0.21 (= 0.7 x 0.3)
C	C	CC	0.09 (= 0.3 x 0.3)

Total probability $= 0.49 + 0.21 + 0.21 + 0.09 = 1$

Exercise 1: Genotype frequency

rs1229984 is a risk variant for various cancers. Freq. of T-allele is 70% and C-allele is 30% in East Asians. If you screened 100 unrelated East Asians, how many would carry the TT, CT and CC genotypes?

From mother	From Father	Genotype	Probability
T	T	TT	0.49 (= 0.7 x 0.7)
C	T	CT	0.21 (= 0.3 x 0.7)
T	C	CT	0.21 (= 0.7 x 0.3)
C	C	CC	0.09 (= 0.3 x 0.3)

Total probability $= 0.49 + 0.21 + 0.21 + 0.09 = 1$

So you should expect

- 49 people to carry the TT allele
- 42 people to carry the CT allele
- 9 people to carry the CC allele

Dice throwing

In an unbiased dice, the prob. of getting a five is $p = 1/6$. If you throw it three times, what is the prob. of observing 2 fives?

Dice throwing

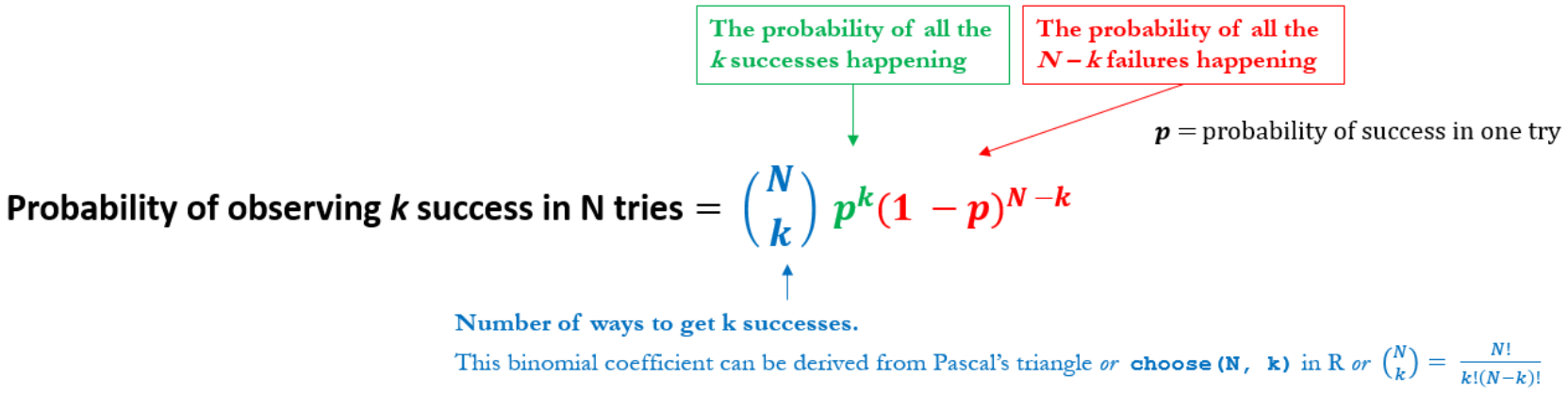
In an unbiased dice, the prob. of getting a five is $p = 1/6$. If you throw it three times, what is the prob. of observing 2 fives?

Throw 1	Throw 2	Throw 3	Number of 5s	Probability
no	no	no	0	$(1-p)^3$
yes	no	no	1	$p \times (1-p)^2$
no	yes	no	1	$p \times (1-p)^2$
no	no	yes	1	$p \times (1-p)^2$
yes	yes	no	2	$p^2 \times (1-p)$
yes	no	yes	2	$p^2 \times (1-p)$
no	yes	yes	2	$p^2 \times (1-p)$
yes	yes	yes	3	p^3

Pr(observe no five) $= (5/6)^3 = 0.579$
Pr(observe 1 five) $= 3 \times (1/6) \times (5/6)^2 = 0.347$
Pr(observe 2 fives) $= 3 \times (1/6)^2 \times (5/6) = 0.069$
Pr(observe 3 fives) $= (1/6)^3 = 0.0046$

Dice throwing | Binomial distribution

The dice throwing example is a demo of the **binomial distribution**:



You can get the probabilities in R:

```
k <- c(0, 1, 2, 3)
dbinom(k, 3, prob=1/6)
## [1] 0.57870370 0.34722222 0.06944444 0.00462963
```

Binomial distribution | Pascal's triangle

$$
\begin{array}{ccc}
\binom{0}{0} & & 1 \\
\binom{1}{0}\binom{1}{1} & & 1 \quad 1 \\
\binom{2}{0}\binom{2}{1}\binom{2}{2} & & 1 \quad 2 \quad 1 \\
\binom{3}{0}\binom{3}{1}\binom{3}{2}\binom{3}{3} & = & 1 \quad 3 \quad 3 \quad 1 \\
\binom{4}{0}\binom{4}{1}\binom{4}{2}\binom{4}{3}\binom{4}{4} & & 1 \quad 4 \quad 6 \quad 4 \quad 1 \\
\binom{5}{0}\binom{5}{1}\binom{5}{2}\binom{5}{3}\binom{5}{4}\binom{5}{5} & & 1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1
\end{array}
$$

Exercise 2: Setup and read in

```
setwd("C:/Users/aramasamy/Desktop/R_workshop")
pacman::p_load(tidyverse, readxl, janitor, data.table)
rm(list=ls())

co2_uptake <- read_csv("data/CO2_Uptake.csv") %>%
  mutate(origin_condition=paste(Type, Treatment))

co2_uptake %>% knitr::kable()
```

Plant	Type	Treatment	conc	uptake	origin_condition
Qn1	Quebec	nonchilled	95	16.0	Quebec nonchilled
Qn1	Quebec	nonchilled	175	30.4	Quebec nonchilled
Qn1	Quebec	nonchilled	250	34.8	Quebec nonchilled
Qn1	Quebec	nonchilled	350	37.2	Quebec nonchilled
Qn1	Quebec	nonchilled	500	35.3	Quebec nonchilled
Qn1	Quebec	nonchilled	675	39.2	Quebec nonchilled
Qn1	Quebec	nonchilled	1000	39.7	Quebec nonchilled

Exercise 2: Setup and read in (alternative)

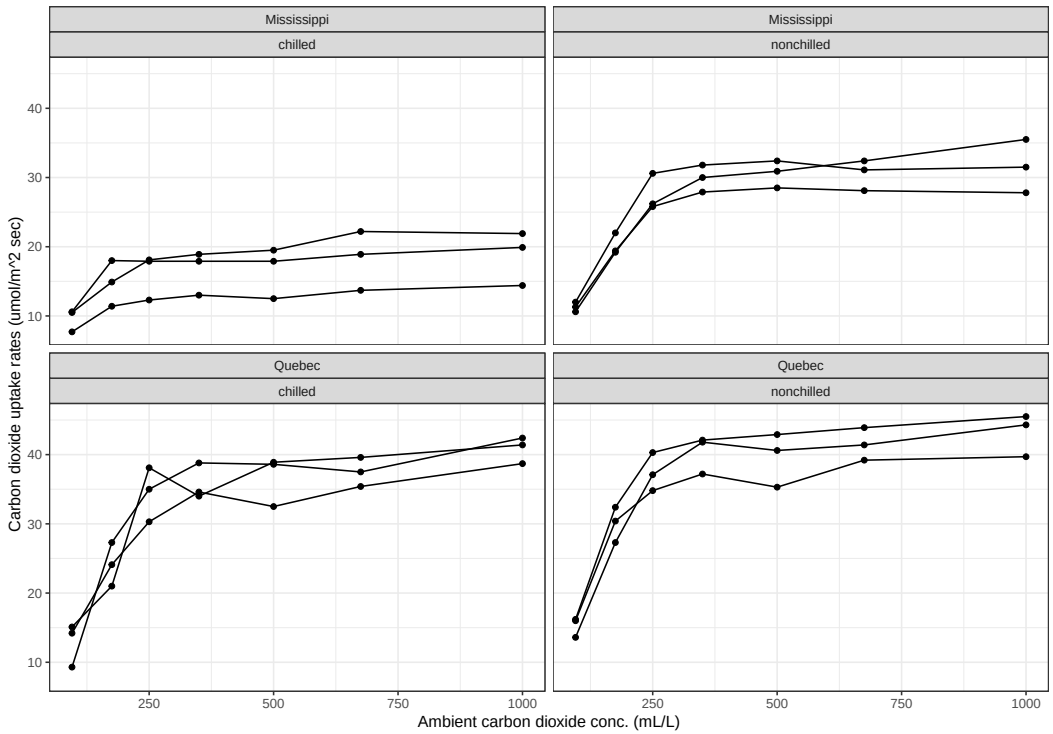
Alternatively, you can also use `unite()`.

```
co2_uptake <- read_csv("data/CO2_Uptake.csv") %>%
  unite("origin_condition", c("Type", "Treatment"), remove=

co2_uptake %>% knitr::kable()
```

Plant	origin_condition	Type	Treatment	conc	uptake
Qn1	Quebec_nonchilled	Quebec	nonchilled	95	16.0
Qn1	Quebec_nonchilled	Quebec	nonchilled	175	30.4
Qn1	Quebec_nonchilled	Quebec	nonchilled	250	34.8
Qn1	Quebec_nonchilled	Quebec	nonchilled	350	37.2
Qn1	Quebec_nonchilled	Quebec	nonchilled	500	35.3
Qn1	Quebec_nonchilled	Quebec	nonchilled	675	39.2
Qn1	Quebec_nonchilled	Quebec	nonchilled	1000	39.7
Qn2	Quebec_nonchilled	Quebec	nonchilled	95	13.6
Qn2	Quebec_nonchilled	Quebec	nonchilled	175	27.3
Qn2	Quebec_nonchilled	Quebec	nonchilled	250	37.1

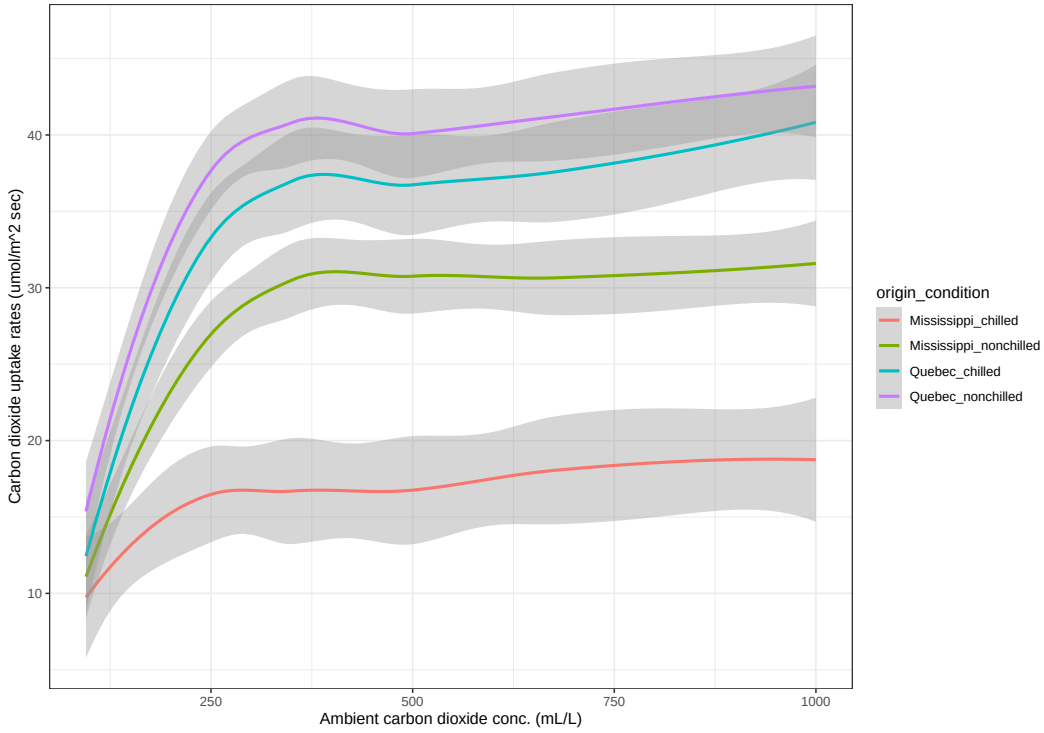
Exercise 2: Show individual plant trajectories (Fig 1)

```
L <- labs(x="Ambient carbon dioxide conc. (mL/L)",
          y="Carbon dioxide uptake rates (umol/m^2 sec)")

ggplot(co2_uptake, aes(x=conc, y=uptake, group=Plant)) +
  geom_point() + geom_line() +
  facet_wrap(Type ~ Treatment) + theme_bw() + L
```



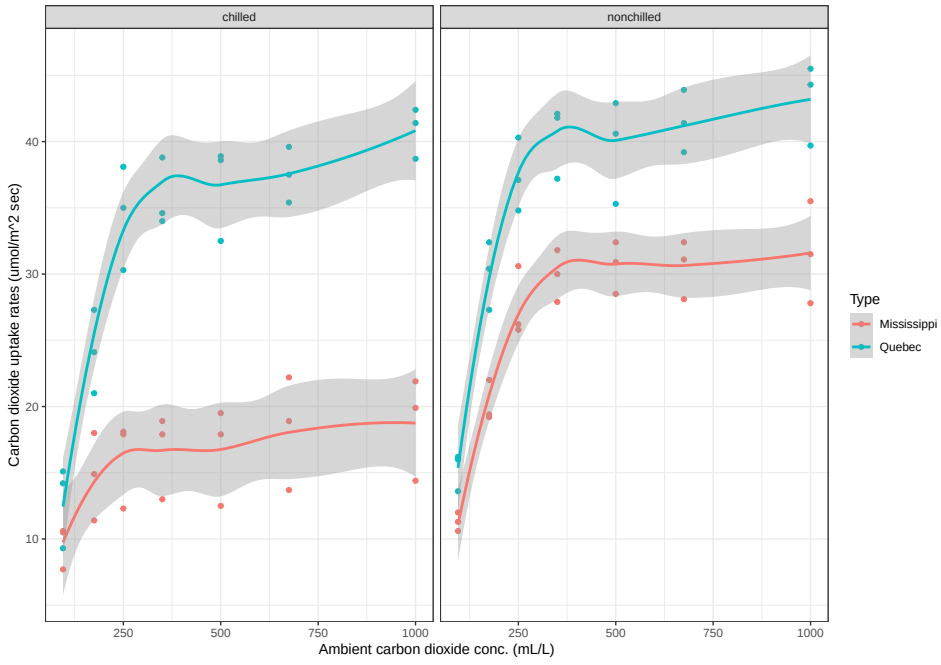
Exercise 2: Smoothed trendline for each group (Fig 2)

```
ggplot(co2_uptake,
       aes(x=conc, y=uptake, col=origin_condition)) +
  geom_smooth() +
  theme_bw() + L
```



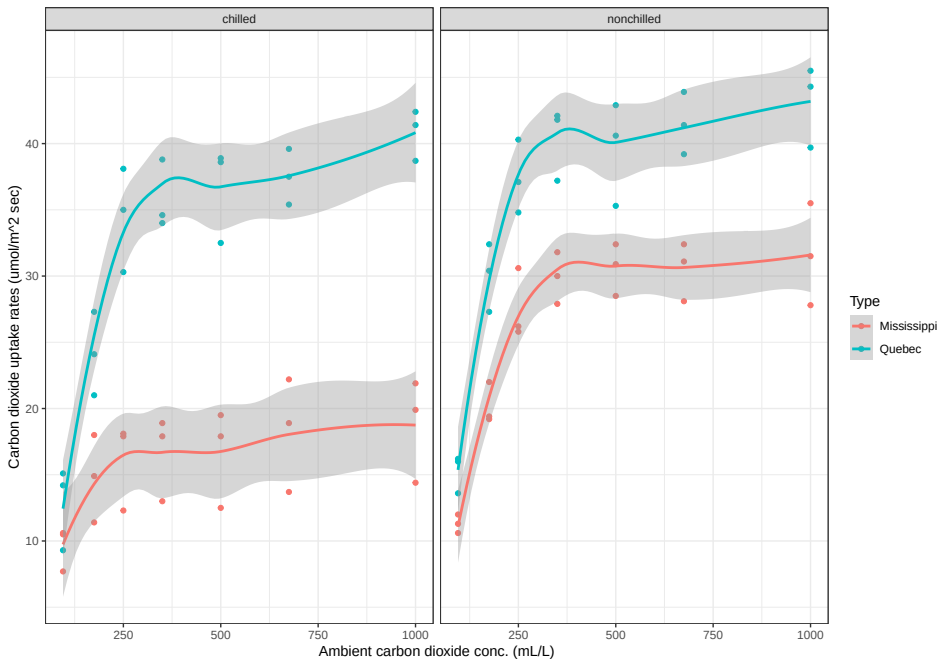
Exercise 2: Smoothed trendline by condition (Fig 3)

```
ggplot(co2_uptake, aes(x=conc, y=uptake, col=Type)) +
  geom_point() + geom_smooth() + facet_wrap(~Treatment) +
  theme_bw() + L
```



Exercise 2: Smoothed trendline by condition (Fig 3)

```
ggplot(co2_uptake, aes(x=conc, y=uptake, col=Type)) +
  geom_point() + geom_smooth() + facet_wrap(~Treatment) +
  theme_bw() + L
```



The CO2 uptake in Quebec plants

1. superior than Mississippi in chilled and non-chilled conditions
2. slightly reduced after chilling (but not as drastic as Misssippi)

Exercise 2: What are the pros and cons of each figure?

Fig 1

- Pros: You can see individual trajectories.
- Cons: Not able to compare across groups. Repetitive.

Fig 2

- Pros: Can compare across groups.
- Cons: Lost individual data points. No story.

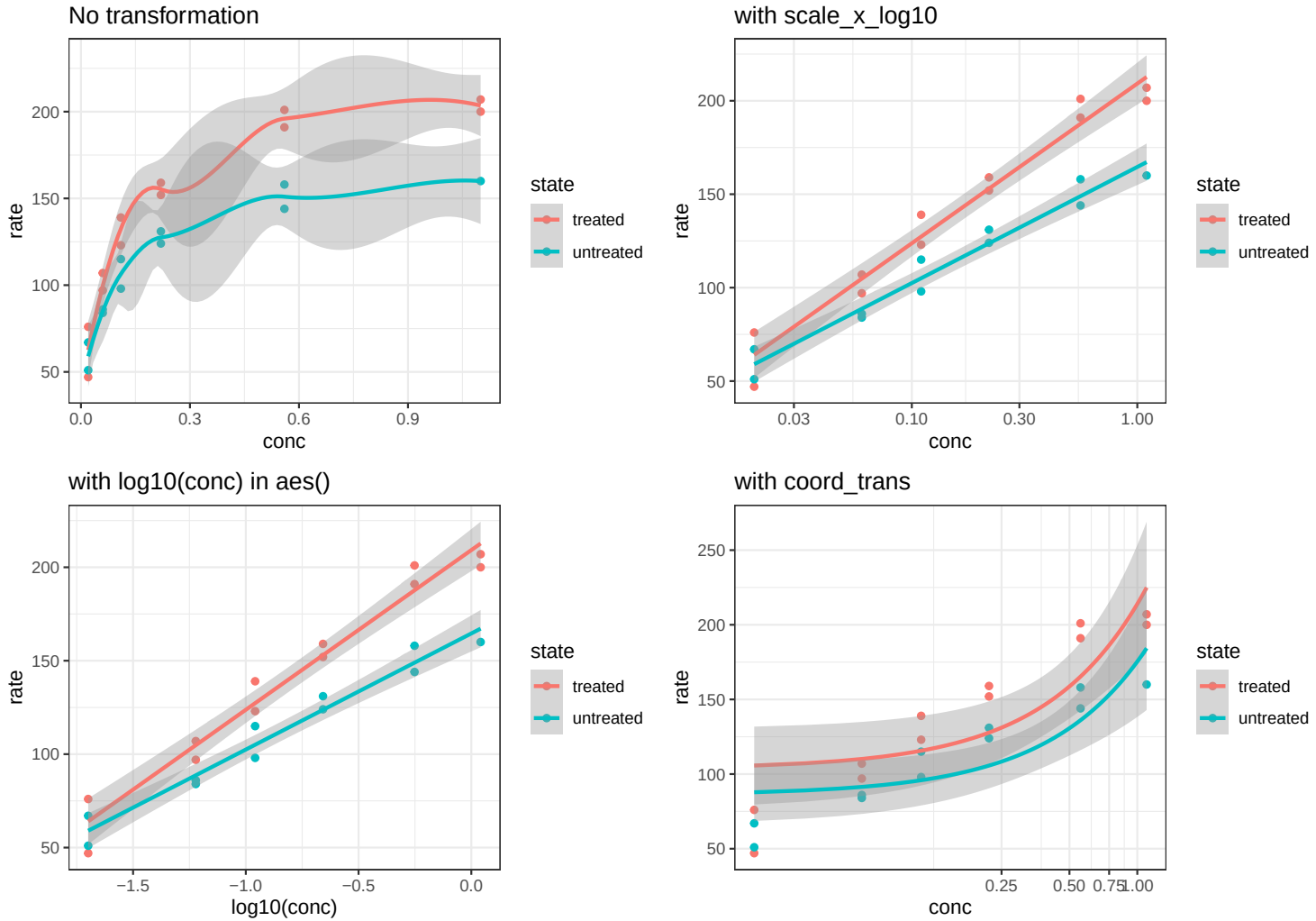
Fig 3

- Pros: Tells a story. Individual data points.
- Cons: Lost individual trajectories.

Exercise 3: Reaction velocity versus substrate conc.

```
Puro <- read_csv("data/puromycin_reaction.csv")

g <- ggplot(Puro, aes(x=conc, y=rate, col=state)) +
       geom_point() + theme_bw()

g1 <- g + geom_smooth() + ggtitle("No transformation")

g2 <- g + geom_smooth(method="lm") + scale_x_log10() +
  ggtitle("with scale_x_log10")    ## Optimal

g3 <- ggplot(Puro, aes(x=log10(conc), y=rate, col=state))+
  geom_point() + geom_smooth(method="lm") + theme_bw() +
  ggtitle("with log10(conc) in aes()")

g4 <- g + geom_smooth(method="lm") +
  coord_trans(x="log10") +
  ggtitle("with coord_trans")
```

Exercise 3: Reaction velocity versus substrate conc.



1) The smooth line on bottom right is not correctly transformed
2) The x-axis on bottom left is difficult to interpret

Note on setup codes

What is not optimal about this code?

```
getwd()
list.files()
setwd("C:/Users/aramasamy/Desktop/R_workshop")
co2 <- read_csv  ("data/CO2_uptake.csv")
install.packages("pacman")
pacman::p_load(tidyverse, readxl, janitor, data.table)
install.packages("grid")
install.packages("gridExtra")
```

Note on setup codes

What is not optimal about this code?

```
getwd()
list.files()
setwd("C:/Users/aramasamy/Desktop/R_workshop")
co2 <- read_csv  ("data/CO2_uptake.csv")
install.packages("pacman")
pacman::p_load(tidyverse, readxl, janitor, data.table)
install.packages("grid")
install.packages("gridExtra")
```

Could be rewritten as

```
setwd("C:/Users/aramasamy/Desktop/R_workshop")
pacman::p_load(tidyverse, readxl, janitor, data.table,
               grid, gridExtra)
rm(list=ls())

co2 <- read_csv("data/CO2_uptake.csv")  # 84 observations
```

Some useful functions

Paste

You can combine two words in one:

```
word1 <- "Hello"
word2 <- "World"
out <- paste(word1, word2)
out
## [1] "Hello World"
```

Paste

You can combine two words in one:

```
word1 <- "Hello"
word2 <- "World"
out <- paste(word1, word2)
out
## [1] "Hello World"
```

Applying `paste()` to two vectors (or two columns in a dataframe), will work element by element:

```
first_part  <- c("blue",  "black", "pink")
second_part <- c("berry", "berry", "guava")
fruit  <- paste(first_part, second_part, sep="")
print(fruit)
## [1] "blueberry"  "blackberry" "pinkguava"
```

Unique

Removes redundant elements leaving only the unique ones.

```
second_part
## [1] "berry" "berry" "guava"

unique(second_part)
## [1] "berry" "guava"
```

Transpose

Swap columns and rows using `t()`

```
tmp <- iris %>% head(3) %>% select(-Species)
tmp
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1          5.1         3.5          1.4         0.2
## 2          4.9         3.0          1.4         0.2
## 3          4.7         3.2          1.3         0.2

t(tmp)
##                    1   2   3
## Sepal.Length 5.1 4.9 4.7
## Sepal.Width  3.5 3.0 3.2
## Petal.Length 1.4 1.4 1.3
## Petal.Width  0.2 0.2 0.2
```

Identical

Checks if two scalars are identical

```
identical( 13, 13 )
## [1] TRUE
identical(  6,  9 )
## [1] FALSE
identical( "5", 5 )
## [1] FALSE
```

Identical

Checks if two scalars are identical

```
identical( 13, 13 )
## [1] TRUE
identical(  6,  9 )
## [1] FALSE
identical( "5", 5 )
## [1] FALSE
```

`identical()` on two vectors tests if both vectors are exactly the same (i.e. no element by element operation like paste):

```
x <- c(1, 2, 3)
y <- c(1, 2, 3)
z <- c(1, 2, 4)
identical(x, y)
## [1] TRUE
identical(y, z)
## [1] FALSE
```

Logical operators

To work on two conditions

```
RIN     <- c(8, 8, 5, 5)
quality <- c("good", "bad",  "good", "bad")

## Option 1 using AND operator
ifelse( RIN > 7 & quality=="good", "pass", "fail" )
## [1] "pass" "fail" "fail" "fail"

## Option 2 using OR operator
ifelse( RIN < 7 | quality=="bad", "fail", "pass" )
## [1] "pass" "fail" "fail" "fail"

## Option 3 using AND and NOT operator
ifelse( !(RIN > 7 & quality=="good"), "fail", "pass" )
## [1] "pass" "fail" "fail" "fail"
```

TCGA breast cancer: clinical data cleanup

Background

We downloaded the clinical and RNA-seq expression data from the Cancer Genome Atlas Program (TCGA) provisional dataset from https://www.cbioportal.org/

The RSEM normalized expression for breast cancer contains 20,531 genes and is 188MB in size.

For this workshop, we restricted the gene expression data to just 6 genes. We also removed the Entrez gene ids.

Step 1: Setup

1. Save the data into R_workshop/data_tcga/ subfolder.
2. Start a new script tcga_classroom.R
3. Set working directory, load packages and remove unwanted objects from work environment
4. Read in the brca_tcga_clinical_data.tsv as `pheno`
5. Explore. How does the colnames look like?

Step 1: Setup

1. Save the data into R_workshop/data_tcga/ subfolder.
2. Start a new script tcga_classroom.R
3. Set working directory, load packages and remove unwanted objects from work environment
4. Read in the brca_tcga_clinical_data.tsv as `pheno`
5. Explore. How does the colnames look like?

```
setwd("C:/Users/aramasamy/Desktop/R_workshop/")
pacman::p_load(tidyverse, janitor)
rm(list=ls())

pheno <- read_tsv("data_tcga/brca_tcga_clinical_data.tsv")
dim(pheno)
## [1] 1108   10
```

Step 2: Cleanup the column names

```
colnames(pheno)
##  [1] "Patient ID"                     "Sample ID"
##  [3] "Cancer Type"                    "Diagnosis Age"
##  [5] "Sex"                            "Ethnicity Category"
##  [7] "Disease Free (Months)"          "Disease Free Status"
##  [9] "Overall Survival (Months)" "Overall Survival Stat
```

Use the `clean_names()` function from the `janitor` package to clean up the column names

Step 2: Cleanup the column names

```
colnames(pheno)
##  [1] "Patient ID"                     "Sample ID"
##  [3] "Cancer Type"                    "Diagnosis Age"
##  [5] "Sex"                            "Ethnicity Category"
##  [7] "Disease Free (Months)"          "Disease Free Status"
##  [9] "Overall Survival (Months)" "Overall Survival Stat
```

Use the `clean_names()` function from the `janitor` package to clean up the column names

```
pheno <- pheno %>% clean_names()

colnames(pheno)
##  [1] "patient_id"                "sample_id"
##  [3] "cancer_type"               "diagnosis_age"
##  [5] "sex"                       "ethnicity_category"
##  [7] "disease_free_months"       "disease_free_status"
##  [9] "overall_survival_months" "overall_survival_status
```

Step 3: Patient ID vs sample ID

1. Which one is more unique?
2. Drop the less useful one.

Step 3: Patient ID vs sample ID

1. Which one is more unique?
2. Drop the less useful one.

```
nrow(pheno)
## [1] 1108

pheno$patient_id %>% unique() %>% length
## [1] 1101

pheno$sample_id  %>% unique() %>% length
## [1] 1108

pheno <- pheno %>% select(-patient_id)
```

Step 4: Tabulate the cancer type

Step 4: Tabulate the cancer type

```
pheno %>% tabyl(cancer_type)
##                cancer_type    n      percent valid_perc
##             Breast Cancer 1093 0.9864620939  0.9909338
##       Breast Cancer, NOS    7 0.0063176895  0.0063463
##           Breast Sarcoma    2 0.0018050542  0.0018132
##  Skin Cancer, Non-Melanoma    1 0.0009025271  0.0009066
##                      <NA>    5 0.0045126354
```

Filter to just the "Breast Cancer". How many samples left?

Step 4: Tabulate the cancer type

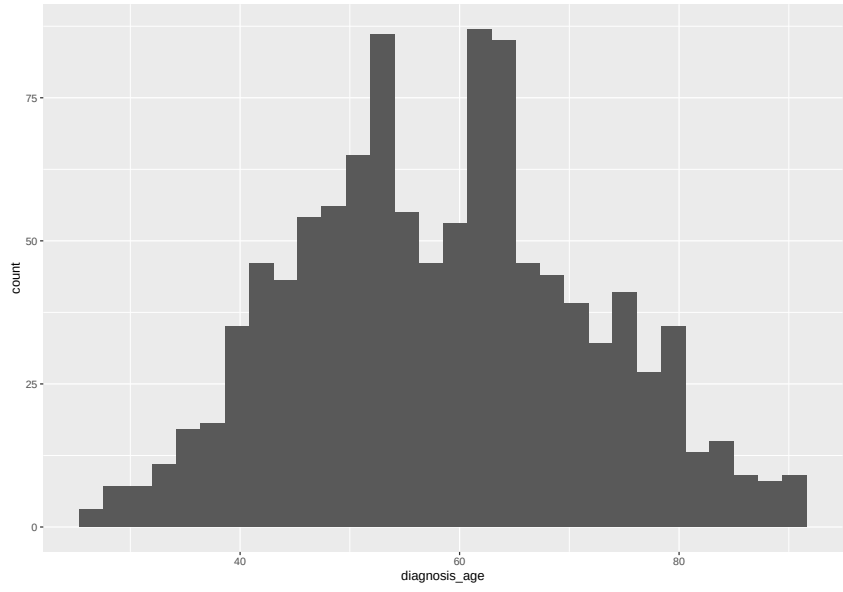
```
pheno %>% tabyl(cancer_type)
##                  cancer_type    n      percent valid_perc
##               Breast Cancer 1093 0.9864620939  0.9909338
##         Breast Cancer, NOS    7 0.0063176895  0.0063463
##             Breast Sarcoma    2 0.0018050542  0.0018132
##  Skin Cancer, Non-Melanoma    1 0.0009025271  0.0009066
##                       <NA>    5 0.0045126354
```

Filter to just the "Breast Cancer". How many samples left?

```
pheno <- pheno %>% filter(cancer_type=="Breast Cancer")
dim(pheno)
## [1] 1093    9
```

Step 5: How does age at diagnosis look like?

```
ggplot(pheno, aes(x=diagnosis_age)) + geom_histogram()
## Warning: Removed 1 rows containing non-finite values (s
```




```
summary(pheno$diagnosis_age)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   26.00   49.00   58.00   58.42   67.00   90.00       1
```

1 missing value in age of diagnosis. Not a concern as we will not be using age at diagnosis very much here.

Step 6: How many females?

Step 6: How many females?

```
pheno %>% tabyl(sex)
##      sex    n    percent
##  Female 1081 0.98902104
##    Male   12 0.01097896
```

Very few male breast cancers. Let's stick to females only. Restrict the data for females only. How many samples left?

Step 6: How many females?

```
pheno %>% tabyl(sex)
##      sex    n    percent
##  Female 1081 0.98902104
##    Male   12 0.01097896
```

Very few male breast cancers. Let's stick to females only. Restrict the data for females only. How many samples left?

```
pheno <- pheno %>% filter(sex=="Female")
dim(pheno)
## [1] 1081    9
```

Step 7: What about ethnicity?

Step 7: What about ethnicity?

```
pheno %>% tabyl(ethnicity_category)
##       ethnicity_category   n    percent valid_percent
##       HISPANIC OR LATINO  37 0.03422757    0.04065934
##  NOT HISPANIC OR LATINO 873 0.80758557    0.95934066
##                    <NA> 171 0.15818686            NA
```

1/6 are not recorded.

Step 7: What about ethnicity?

```
pheno %>% tabyl(ethnicity_category)
##       ethnicity_category   n    percent valid_percent
##       HISPANIC OR LATINO  37 0.03422757    0.04065934
##  NOT HISPANIC OR LATINO 873 0.80758557    0.95934066
##                    <NA> 171 0.15818686            NA
```

1/6 are not recorded.

Rename ethnicity_category as ethnicity

```
pheno <- pheno %>% rename(ethnicity=ethnicity_category)
```

Step 8: Disease free survival

1. Rename disease_free_months to DFS_months
2. Check distribution of DFS_months. You may find `summary()` and `sort()` useful

Step 8: Disease free survival

1. Rename disease_free_months to DFS_months
2. Check distribution of DFS_months. You may find `summary()` and `sort()` useful

```
pheno <- pheno %>% rename(DFS_months=disease_free_months)

summary(pheno$DFS_months)
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##    -0.23    14.16    24.70    37.21    50.33   281.08       92

pheno$DFS_months %>% sort %>% head(20)
##  [1] -0.23  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0
## [12]  0.00  0.00  0.03  0.03  0.16  0.16  0.16  0.23  0
```

DFS is negative for 1 sample, zero for 12 samples and missing for another 93.

Step 8: Disease free survival

3. Restrict to only samples with positive DFS. How many samples left?

Step 8: Disease free survival

3. Restrict to only samples with positive DFS. How many samples left?

```
pheno <- pheno %>% filter(DFS_months > 0)
dim(pheno)  # 976
## [1] 976   9
```

Step 9: Disease free status

1. Tabulate the disease free status
2. Create a new column called “event”. Set to 0 if they are disease free. Otherwise set to 1. Tabulate to confirm.
3. Drop the disease_free_status column.

Step 9: Disease free status

1. Tabulate the disease free status
2. Create a new column called "event". Set to 0 if they are disease free. Otherwise set to 1. Tabulate to confirm.
3. Drop the disease_free_status column.

```
pheno %>% tabyl(disease_free_status)
##  disease_free_status   n   percent
##          DiseaseFree 865 0.8862705
##  Recurred/Progressed 111 0.1137295

pheno <- pheno %>%
  rename(DFS=disease_free_status) %>%
  mutate(event=ifelse(DFS=="DiseaseFree", 0, 1)) %>%
  select(-DFS)

pheno %>% tabyl(event)
##  event   n   percent
##      0 865 0.8862705
##      1 111 0.1137295
```

Step 10: Overall survival

The overall survival is not very useful as reason for death is not captured here.

Drop the overall survival data. How many samples do you have?

Step 10: Overall survival

The overall survival is not very useful as reason for death is not captured here.

Drop the overall survival data. How many samples do you have?

```
pheno <- pheno %>% select(-starts_with("overall"))
dim(pheno)
## [1] 976   7
```

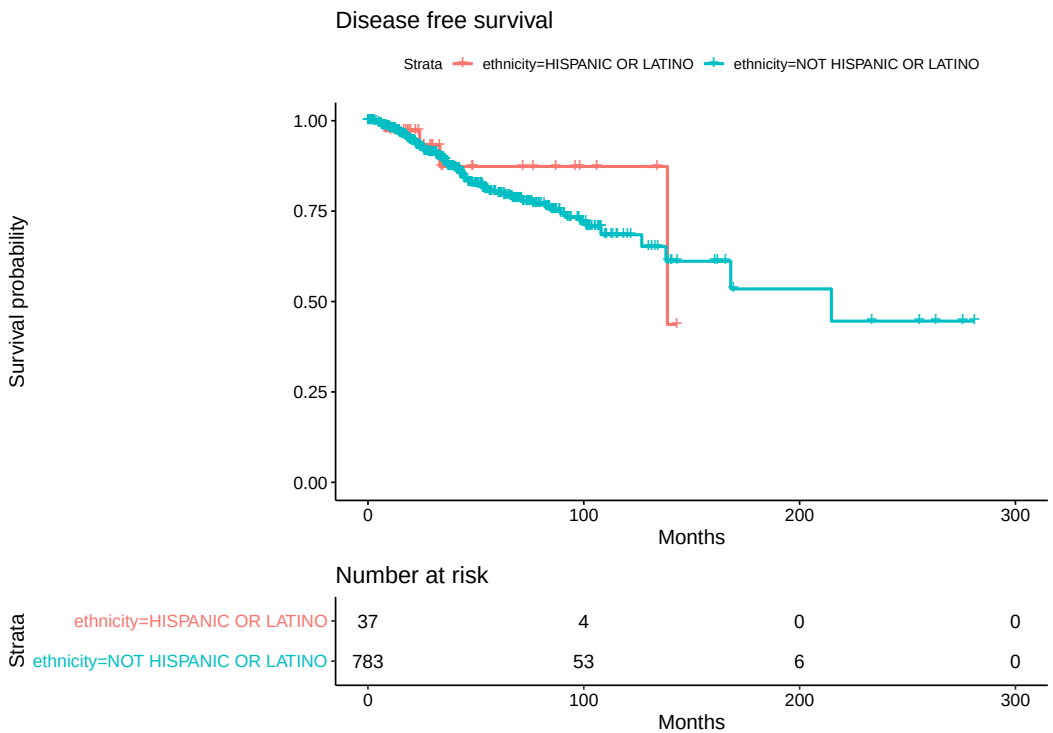
Aside: Kaplan-Meir curves

```
pacman::p_load(survminer, survival)

pheno$DFS <- Surv(pheno$DFS_months, pheno$event)

tail(pheno$DFS_months)
## [1]  52.92  29.01  15.34  16.03 107.98 106.96

tail(pheno$event)
## [1] 0 0 0 0 1 0

tail(pheno$DFS)
## [1]  52.92+  29.01+  15.34+  16.03+ 107.98  106.96+
```

Note: Using `Surv()` inside `mutate()` appears to be incompatible.

Aside: Kaplan-Meir curves

```
fit <- survfit( DFS ~ ethnicity, data=pheno)

ggsurvplot(fit, risk.table=TRUE,
           xlab="Months",
           title="Disease free survival")
```



Warning: We ignored tumor stage, type, age, treatment etc here.

TCGA breast cancer: expression data cleanup

Step 1: Read in the expression data

1. Read in the expression data as `expr`.
2. Set Hugo_Symbol as rownames using `column_to_rownames()` function.
3. How many samples are there?

Step 1: Read in the expression data

1. Read in the expression data as `expr`.
2. Set Hugo_Symbol as rownames using `column_to_rownames()` function.
3. How many samples are there?

```
fn   <- "data_tcga/brca_RNA_Seq_v2_expression_median_sel.tx
expr <- read_delim(fn, delim="\t")
expr <- expr %>% column_to_rownames("Hugo_Symbol")
dim(expr)
## [1]    6 1100

expr[ , 1:3]
##        TCGA-3C-AAAU-01 TCGA-3C-AALI-01 TCGA-3C-AALJ-01
## CHFR          359.4792       1100.0544        878.5131
## CXCL13         19.6456        529.6357        384.4062
## EN1            26.5387         40.7830        116.9538
## FOXJ1          13.7864          5.9815         19.9456
## GAPDH       28979.6031      80417.0745      59057.1170
## TAT            14.1310        274.6058        643.6990
```

Step 2: Transpose the data

Each row in `pheno` represents a sample.

Each column in `expr` represents a sample.

Transpose the expression data so that the rows represent samples in `expr`.

Step 2: Transpose the data

Each row in `pheno` represents a sample.

Each column in `expr` represents a sample.

Transpose the expression data so that the rows represent samples in `expr`.

```
expr <- t(expr)

dim(expr)
## [1] 1100    6

head(expr, 3) %>% round(1)
##                    CHFR CXCL13   EN1 FOXJ1   GAPDH   TAT
## TCGA-3C-AAAU-01  359.5   19.6  26.5  13.8 28979.6  14.1
## TCGA-3C-AALI-01 1100.1  529.6  40.8   6.0 80417.1 274.6
## TCGA-3C-AALJ-01  878.5  384.4 117.0  19.9 59057.1 643.7
```

Step 3: Log2 transformation

We need transform gene expression data so that it suitable for statistical modelling later.

Several methods exist to do this. The simplest is log2() transformation.

```
expr <- log2( expr + 1 )
head(expr, 3) %>% round(1)
##                    CHFR CXCL13 EN1 FOXJ1 GAPDH TAT
## TCGA-3C-AAAU-01  8.5    4.4 4.8   3.9  14.8 3.9
## TCGA-3C-AALI-01 10.1    9.1 5.4   2.8  16.3 8.1
## TCGA-3C-AALJ-01  9.8    8.6 6.9   4.4  15.8 9.3
```

Note: We need to add 1 as some expression values are 0 and log2(0) is undefined.

Upcoming tasks

```
dim(pheno)
## [1] 976   8

dim(expr)
## [1] 1100    6

identical( pheno$sample_id, rownames(expr) )
## [1] FALSE
```

1. Which samples are represented in the clinical dataset only, expression data only and in both?
2. Combine the clinical data and expression together.

Set Operators

Set Operators | Fruit example

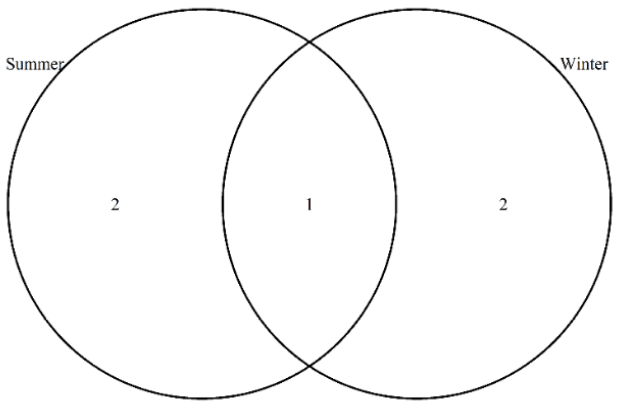
```
summer <- c("apple", "cherries", "watermelon")
winter <- c("apple", "oranges", "lemons")
```

Which fruits are found

1. in both summer and winter?
2. in summer only?
3. in winter only?
4. in this universe?

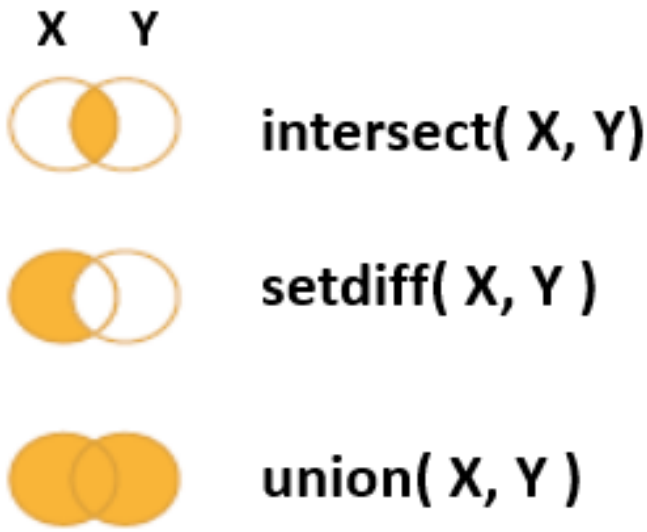
Set Operators | Venn diagram

```
pacman::p_load(VennDiagram)

summer <- c("apple", "cherries", "watermelon")
winter <- c("apple", "oranges", "lemons")

venn.diagram(list(summer, winter),
             category.names=c("Summer", "Winter"),
             filename="venn.png")
```



Complementary tool: https://bioinfogp.cnb.csic.es/tools/venny/

Set Operators | Venn diagram terminology



Set Operators | intersect(), setdiff(), union()

```r
summer <- c("apple", "cherries", "watermelon")
winter <- c("apple", "oranges", "lemons")

intersect(summer, winter)  ## In both
## [1] "apple"

setdiff(summer, winter)    ## In summer only
## [1] "cherries"   "watermelon"

setdiff(winter, summer)    ## In winter only
## [1] "oranges" "lemons"

union(summer, winter)      ## In the universe
## [1] "apple"      "cherries"   "watermelon" "oranges"
```

Set Operators | TCGA dataset

How many samples are found

- only the clinical/phenotype file
- only the expression file
- common to both file

Set Operators | TCGA dataset

How many samples are found

- only the clinical/phenotype file
- only the expression file
- common to both file

```
x <- pheno$sample_id;  y <- rownames(expr)

# Found only in the phenotype file
setdiff(x, y) %>% length()
## [1] 3

# Found only in the expression file
setdiff(y, x) %>% length()
## [1] 127

# Found in both
intersect(x, y) %>% length()
## [1] 973
```

Combining multiple datasets

Row binding

```
first <- head(iris, 1) %>% select(Species, Sepal.Length)
first
##   Species Sepal.Length
## 1  setosa          5.1

last  <- tail(iris, 1) %>% select(Species, Sepal.Length)
last
##       Species Sepal.Length
## 150 virginica          5.9
```

Row binding

```
first <- head(iris, 1) %>% select(Species, Sepal.Length)
first
##   Species Sepal.Length
## 1  setosa          5.1

last  <- tail(iris, 1) %>% select(Species, Sepal.Length)
last
##       Species Sepal.Length
## 150 virginica          5.9
```

```
bind_rows(first, last)
##     Species Sepal.Length Petal.Length
## 1    setosa          5.1           NA
## 2 virginica           NA          5.1
```

Row binding when column names are not matching

```
first <- head(iris, 1) %>% select(Species, Sepal.Length)
first
##   Species Sepal.Length
## 1  setosa          5.1

last  <- tail(iris, 1) %>% select(Species, Petal.Length)
last
##       Species Petal.Length
## 150 virginica          5.1
```

Row binding when column names are not matching

```
first <- head(iris, 1) %>% select(Species, Sepal.Length)
first
##   Species Sepal.Length
## 1  setosa          5.1

last  <- tail(iris, 1) %>% select(Species, Petal.Length)
last
##       Species Petal.Length
## 150 virginica          5.1
```

```
bind_rows(first, last)
##     Species Sepal.Length Petal.Length
## 1    setosa          5.1           NA
## 2 virginica           NA          5.1
```

When row-binding, columns are matched by name, and any missing columns will be filled with NA.

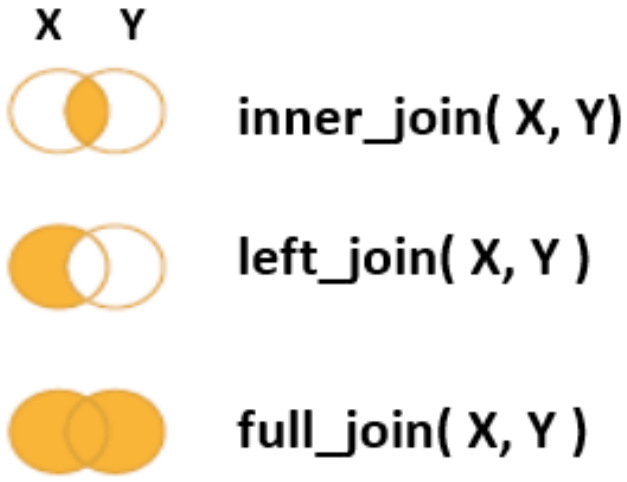
Column binding

The function bind_cols() matches by position, so both datasets must be in same order. **Risky!**.

We will show you join() which matches using keys instead.

join | Terminology

join | Terminology



Note:

1) *join functions works on dataframes (2-dimensional data). Also can use multiple keys.

2) intersect/setdiff/unions work on vectors (1-dimensional data)

join | Example dataset

```
data("band_members")
band_members
## name band
## 1 Mick Stones
## 2 John Beatles
## 3 Paul Beatles

data("band_instruments")
band_instruments
## name plays
## 1 John guitar
## 2 Paul bass
## 3 Keith guitar
```

Which columns are common across both dataset? This is your "key".

`*_join()` family to merge 2 files

Must have at least one common key across both files:

*_join() family to merge 2 files

Must have at least one common key across both files:

*join()

join() merges data based on common column names

```
> data("band_members")
> band_members
# A tibble: 3 x 2
  name  band
  <chr> <chr>
1 Mick  Stones
2 John  Beatles
3 Paul  Beatles
```

```
> data("band_instruments")
> band_instruments
# A tibble: 3 x 2
  name  plays
  <chr> <chr>
1 John  guitar
2 Paul  bass
3 Keith guitar
```

```
> band_members %>% inner_join(band_instruments)
Joining, by = "name"
# A tibble: 2 x 3
  name  band    plays
  <chr> <chr>   <chr>
1 John  Beatles guitar
2 Paul  Beatles bass
```

Important to keep an eye on this as it tells you which keys it is joining on

```
> band_members %>% left_join(band_instruments)
Joining, by = "name"
# A tibble: 3 x 3
  name  band    plays
  <chr> <chr>   <chr>
1 Mick  Stones  NA
2 John  Beatles guitar
3 Paul  Beatles bass
```

```
> band_members %>% right_join(band_instruments)
Joining, by = "name"
# A tibble: 3 x 3
  name  band    plays
  <chr> <chr>   <chr>
1 John  Beatles guitar
2 Paul  Beatles bass
3 Keith NA      guitar
```

```
> band_members %>% full_join(band_instruments)
Joining, by = "name"
# A tibble: 4 x 3
  name  band    plays
  <chr> <chr>   <chr>
1 Mick  Stones  NA
2 John  Beatles guitar
3 Paul  Beatles bass
4 Keith NA      guitar
```

join_all() to merge more that 2 files

Must have at least one common key across **all** files:

```
> ## Create some fake data for illustration
> df1 <- data.frame( ENSGID=c("ENSG05339", "ENSG05513", "ENSG08128"),
+                    SYMBOL=c("CREBBP", "SOX8", "CDK11A"),
+                    count1=c(340, 5000, 20) )
>
> df2 <- data.frame( ENSGID=c("ENSG05513", "ENSG08128", "ENSG34971"),
+                    SYMBOL=c("SOX8", "CDK11A", "MYOC"),
+                    count2=c(4500, 6, 250) )
>
> df3 <- data.frame( ENSGID=c("ENSG05339", "ENSG08128", "ENSG34971"),
+                    SYMBOL=c("CREBBP", "CDK11A", "MYOC"),
+                    count3=c(1000, 25, 400),
+                    batch3=c(4500, 6, 250))
>
> ## View the data
> df1
     ENSGID SYMBOL count1
1 ENSG05339 CREBBP    340
2 ENSG05513   SOX8   5000
3 ENSG08128 CDK11A     20
> df2
     ENSGID SYMBOL count2
1 ENSG05513   SOX8   4500
2 ENSG08128 CDK11A      6
3 ENSG34971   MYOC    250
> df3
     ENSGID SYMBOL count3 batch3
1 ENSG05339 CREBBP   1000   4500
2 ENSG08128 CDK11A     25      6
3 ENSG34971   MYOC    400    250
```

```
> ## Full or outer join using the universe of the keys
> ## Here, ENSGID and SYMBOL are both used as keys
>
> library(plyr)
> join_all( list(df1, df2,df3), type="full")
Joining by: ENSGID, SYMBOL
Joining by: ENSGID, SYMBOL
     ENSGID SYMBOL count1 count2 count3 batch3
1 ENSG05339 CREBBP    340     NA   1000   4500
2 ENSG05513   SOX8   5000   4500     NA     NA
3 ENSG08128 CDK11A     20      6     25      6
4 ENSG34971   MYOC     NA    250    400    250
> ## Notice:
> #1. The value for the missing keys is padded as NA
> #2. Merges all the columns that have the same column name.
>
> ## You can easily extend to more than two files
> ## by adding into the list() argument above
```

join | TCGA dataset

Join the clinical and expression data together keeping only samples with both information.

join | TCGA dataset

Join the clinical and expression data together keeping only samples with both information.

Step 1: Join only works with data frames. Let's convert `expr` to a data frame first.

```
expr <- data.frame(expr) %>%
  rownames_to_column("sample_id")

head(expr)
##          sample_id      CHFR    CXCL13      EN1    FOXJ1
## 1 TCGA-3C-AAAU-01  8.493772  4.367762 4.783389 3.886199
## 2 TCGA-3C-AALI-01 10.104670  9.051578 5.384844 2.803537
## 3 TCGA-3C-AALJ-01  9.780561  8.590236 6.882078 4.388575
## 4 TCGA-3C-AALK-01  9.312137  6.052466 5.959606 9.361065
## 5 TCGA-4H-AAAK-01  8.912444  5.940766 6.269972 3.041593
## 6 TCGA-5L-AAT0-01  8.978137 10.107497 5.701380 4.073657
```

join | TCGA dataset

Step 2: Use inner_join to keep only samples present in both

```
comb <- inner_join(pheno, expr)

colnames(comb)
##  [1] "sample_id"      "cancer_type"    "diagnosis_age" "s
##  [5] "ethnicity"      "DFS_months"     "event"         "D
##  [9] "CHFR"           "CXCL13"         "EN1"           "F
## [13] "GAPDH"          "TAT"

head(comb, 3)
## # A tibble: 3 x 14
##   sample_id cancer_type diagnosis_age sex   ethnicity D
##   <chr>     <chr>                <dbl> <chr> <chr>
## 1 TCGA-3C-~ Breast Can~             50 Fema~ NOT HISP~
## 2 TCGA-3C-~ Breast Can~             62 Fema~ NOT HISP~
## 3 TCGA-3C-~ Breast Can~             52 Fema~ NOT HISP~
## # ... with 8 more variables: DFS[,"time"] <dbl>, [,"sta
## #   CHFR <dbl>, CXCL13 <dbl>, EN1 <dbl>, FOXJ1 <dbl>, C
```