R COOKBOOK – TIDY DATA IN R

Robbie Stancil – CISER GRA

Session Info

- Today we’ll discuss tidying and manipulating entire data sets R
- Want to learn even more? Join in on another session:
  - 3/24 – Data Visualization using ggplot2
  - 3/31 – Reproducible Reports using RMarkdown
- Have questions? Feel free to interrupt and ask!

Defining Tidy Data

- In Tidy Data:
  1. Every column is a variable
  2. Every row is an observation
  3. Every Cell is a single value

https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

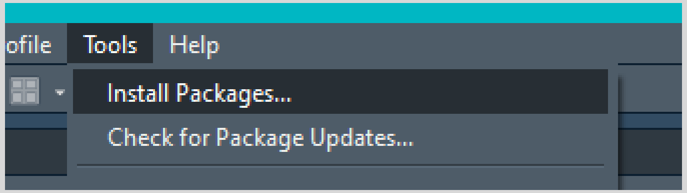
Tidyverse

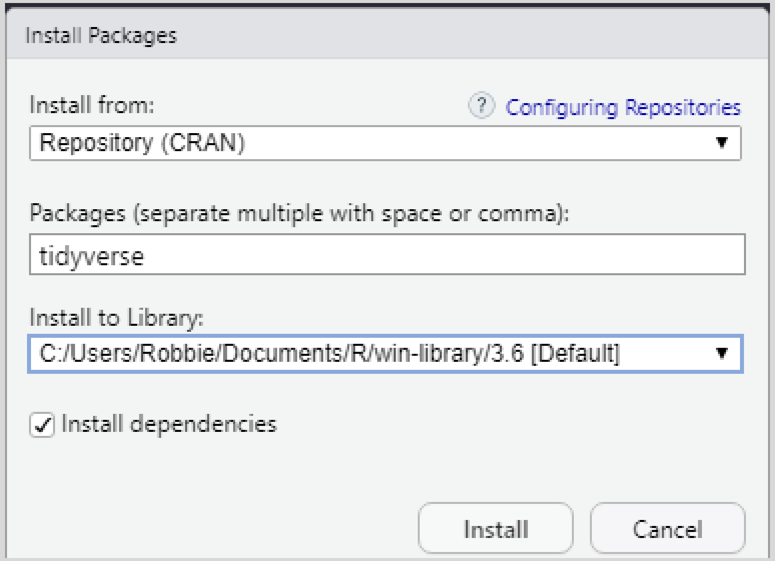
- Collection of 8 main packages designed for data science including dplyr, tibble, and more
- Expands on the default data frame structure in R
- Provides alternative ways to manipulate and work with data

https://www.tidyverse.org/

Installing Tidyverse

- Method 1:





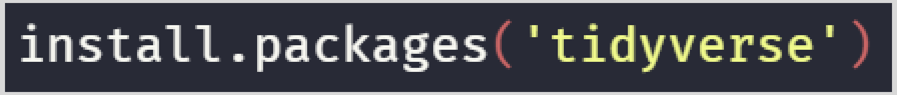
Install from: Repository (CRAN)

Packages (separate multiple with space or comma): tidyverse

Install to Library: C:/Users/Robbie/Documents/R/win-library/3.6 [Default]

☑ Install dependencies

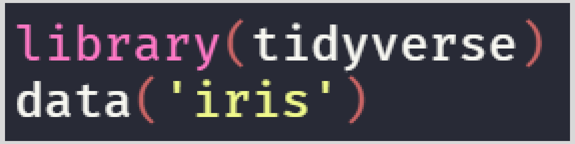
- Method 2:



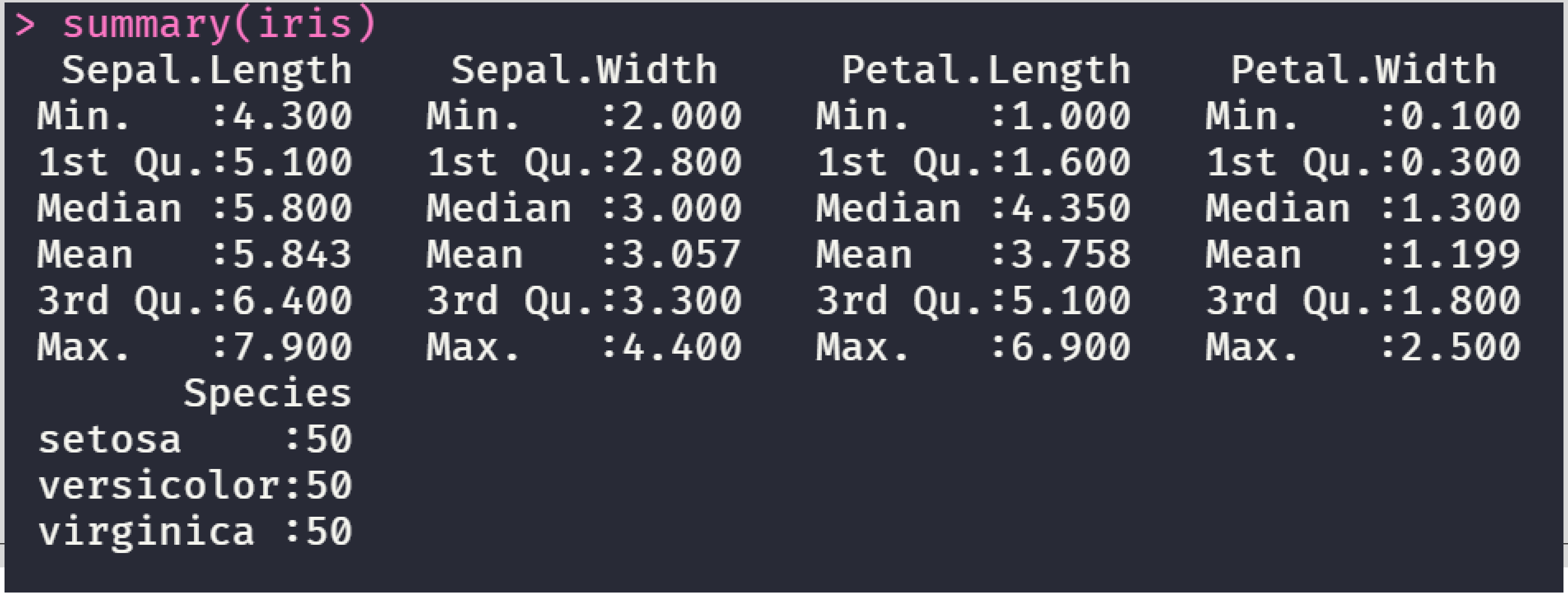
```
install.packages('tidyverse')
```

Preparing RStudio

- Import the tidyverse package
- Load our data



```
library(tidyverse)
data('iris')
```



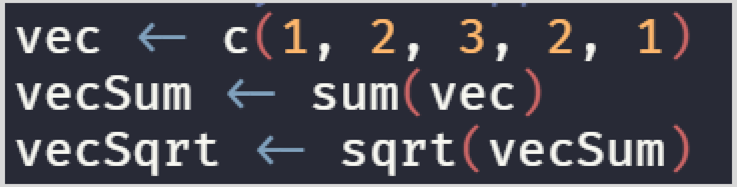
```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

Pipe Operator

- %>% (keyboard shortcut Ctrl+Shift+M)
- Allows you to take the output from one function and pass it to another function
- Simplifies code and improves flow and readability

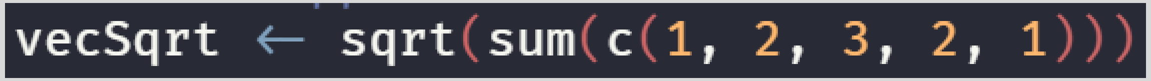
Advantages of Piping

- Line by line approach:



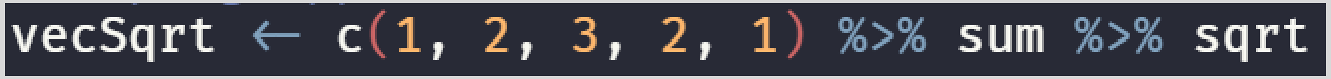
```
vec ← c(1, 2, 3, 2, 1)
vecSum ← sum(vec)
vecSqrt ← sqrt(vecSum)
```

- Nested approach:



```
vecSqrt ← sqrt(sum(c(1, 2, 3, 2, 1)))
```

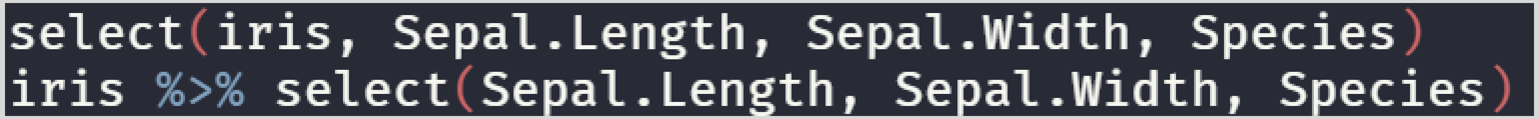
- Piping approach:



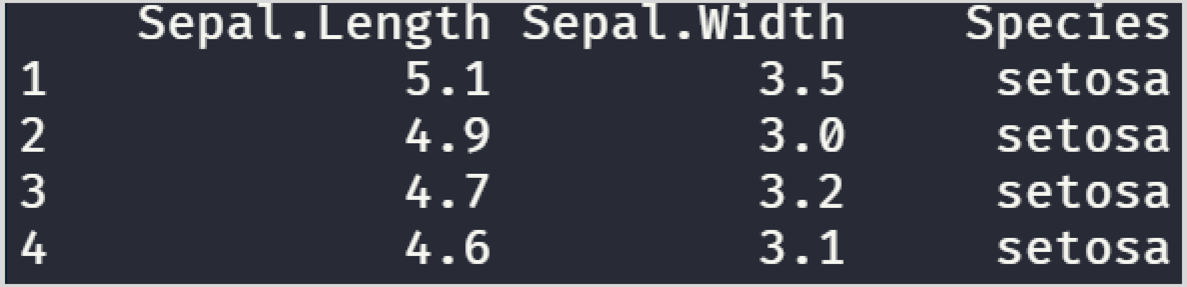
```
vecSqrt ← c(1, 2, 3, 2, 1) %>% sum %>% sqrt
```

select()

- Select columns from the data
- Input: our data frame and a list of columns we want



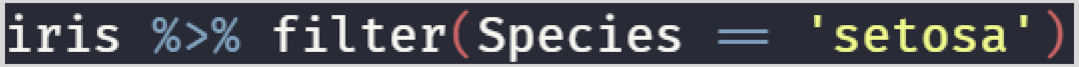
```
select(iris, Sepal.Length, Sepal.Width, Species)
iris %>% select(Sepal.Length, Sepal.Width, Species)
```



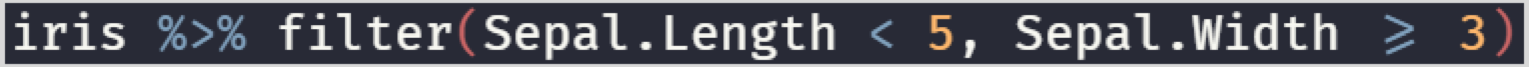
```
    Sepal.Length Sepal.Width    Species
1            5.1         3.5     setosa
2            4.9         3.0     setosa
3            4.7         3.2     setosa
4            4.6         3.1     setosa
```

filter()

- Only display rows that meet some requirement(s)
- Input: our data frame and a relational expression
  - Examples: >, <, >=, <=, ==, !=, is.na()

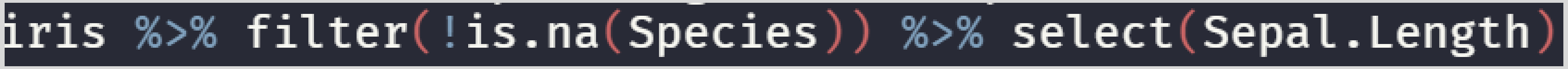


```r
iris %>% filter(Species == 'setosa')
```

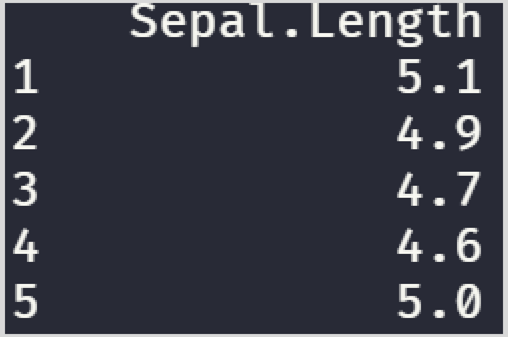


```r
iris %>% filter(Sepal.Length < 5, Sepal.Width >= 3)
```

filter()



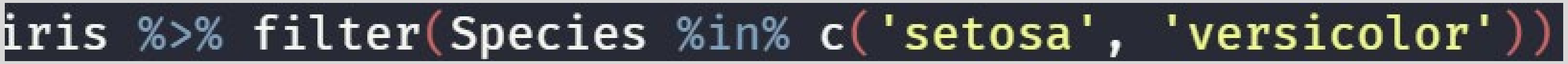
```
iris %>% filter(!is.na(Species)) %>% select(Sepal.Length)
```



```
  Sepal.Length
1          5.1
2          4.9
3          4.7
4          4.6
5          5.0
```

One Last Relational Expression

- %in%
- Check a column for multiple values



```
iris %>% filter(Species %in% c('setosa', 'versicolor'))
```

group_by() and summarize()

- group_by()
  - groups data based on the value of a column or columns
  - Needs to be used in combination with summarize to get the full use
  - Mostly used on categorical values

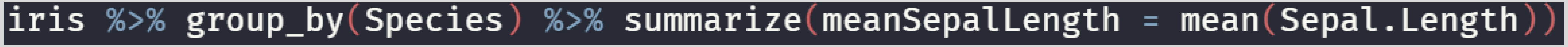
group_by() and summarize()

- Summarize
  - Computes specified statistics over the given data grouping
  - Will compute the statistic over the last grouping
  - Can compute as many values at once as you would like
  - Input: our data frame followed by the following:
    - <name> = statistic

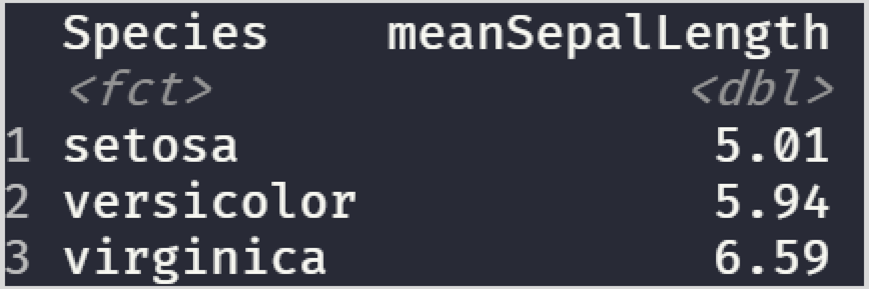
group_by() and summarize()

- Example Statistics:
  - mean()
  - sd()
  - max()
  - min()
  - n()

group_by() and summarize()

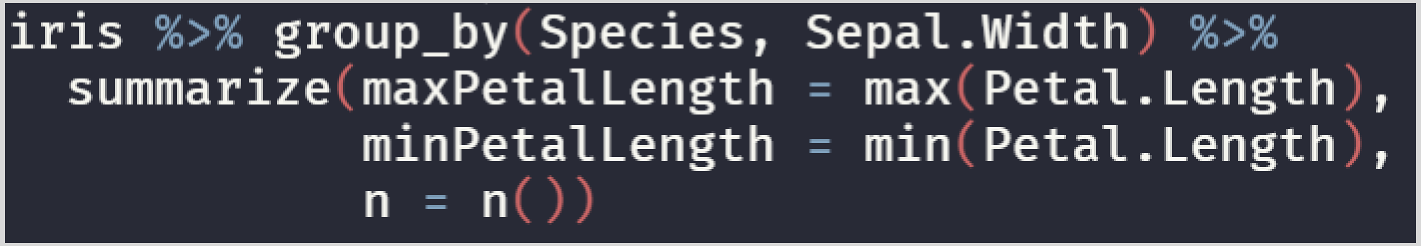


```
iris %>% group_by(Species) %>% summarize(meanSepalLength = mean(Sepal.Length))
```

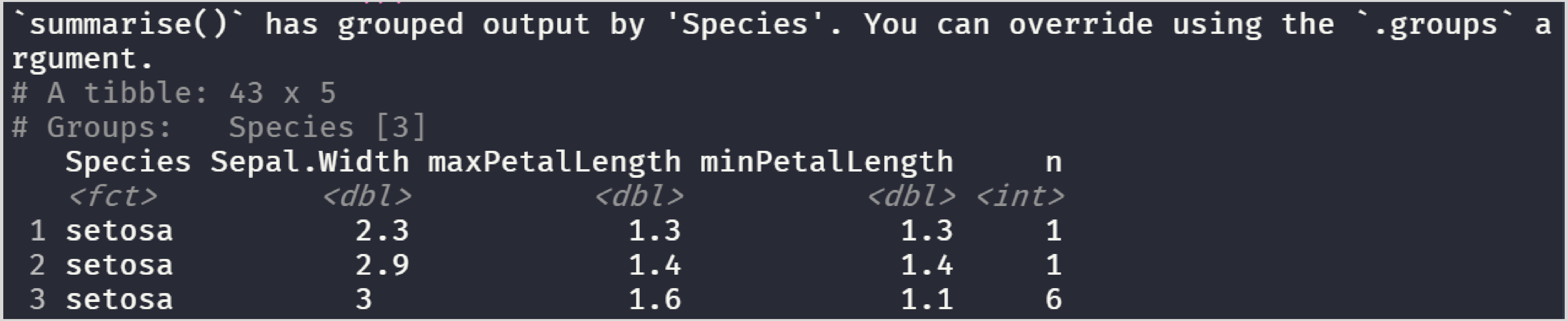


```
  Species    meanSepalLength
  <fct>                <dbl>
1 setosa                5.01
2 versicolor            5.94
3 virginica             6.59
```

group_by() and summarize()

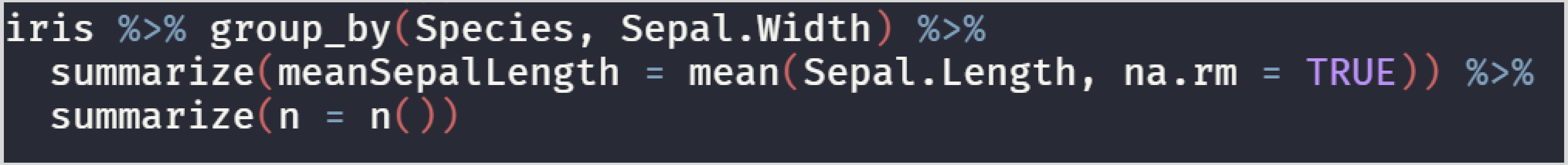


```
iris %>% group_by(Species, Sepal.Width) %>%
  summarize(maxPetalLength = max(Petal.Length),
            minPetalLength = min(Petal.Length),
            n = n())
```

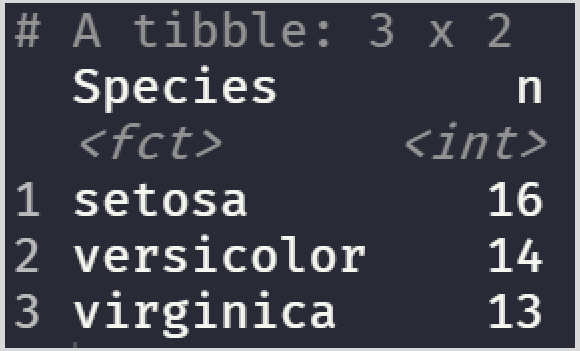


```
`summarise()` has grouped output by 'Species'. You can override using the `.groups` a
rgument.
# A tibble: 43 x 5
# Groups:   Species [3]
   Species Sepal.Width maxPetalLength minPetalLength     n
   <fct>         <dbl>          <dbl>          <dbl> <int>
 1 setosa          2.3            1.3            1.3     1
 2 setosa          2.9            1.4            1.4     1
 3 setosa          3              1.6            1.1     6
```

group_by() and summarize()



```
iris %>% group_by(Species, Sepal.Width) %>%
  summarize(meanSepalLength = mean(Sepal.Length, na.rm = TRUE)) %>%
  summarize(n = n())
```

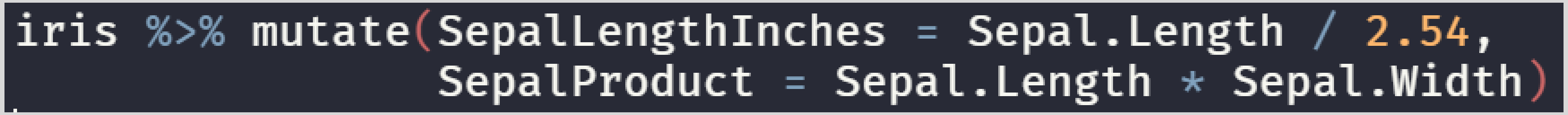


```
# A tibble: 3 x 2
  Species        n
  <fct>      <int>
1 setosa        16
2 versicolor    14
3 virginica     13
```

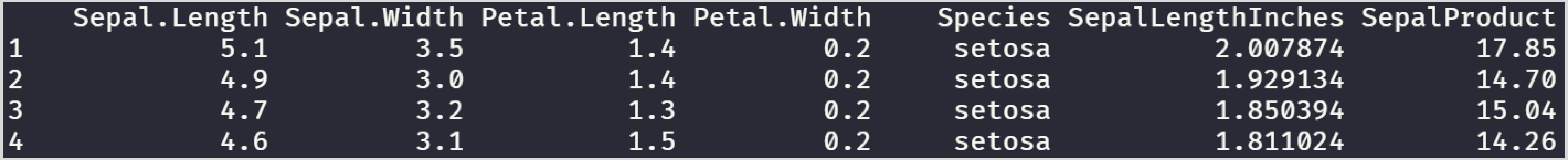
mutate()

- Creates a new column
- Input: our data frame followed by the following:
  - <new column name> = value

mutate()

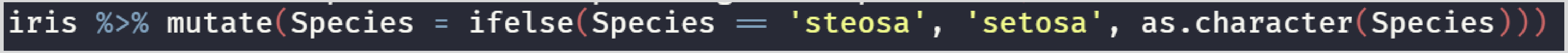


```
iris %>% mutate(SepalLengthInches = Sepal.Length / 2.54,
                SepalProduct = Sepal.Length * Sepal.Width)
```

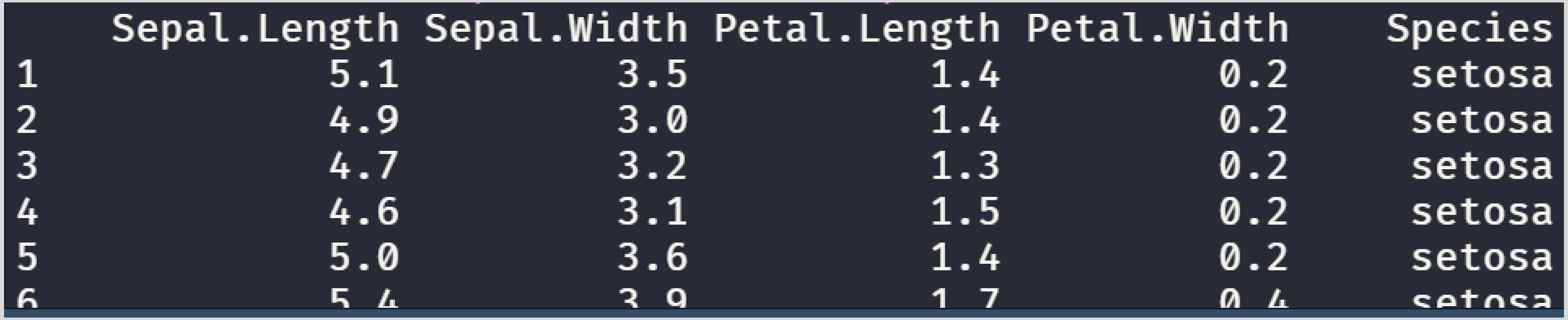


```
  Sepal.Length Sepal.Width Petal.Length Petal.Width    Species SepalLengthInches SepalProduct
1          5.1         3.5          1.4         0.2     setosa          2.007874        17.85
2          4.9         3.0          1.4         0.2     setosa          1.929134        14.70
3          4.7         3.2          1.3         0.2     setosa          1.850394        15.04
4          4.6         3.1          1.5         0.2     setosa          1.811024        14.26
```

mutate()



```
iris %>% mutate(Species = ifelse(Species == 'steosa', 'setosa', as.character(Species)))
```



```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
```

Ending Notes

- Thanks for joining in – if you have any questions please ask!
- 3/24 – Data Visualization using ggplot2
- 3/31 – Reproducible Reports using RMarkdown