About me

Scholarly Communications Librarian at the University of Houston-Clear Lake
Outline for Today

• Cleaning up strings with stringr
• Grouping & summarizing data with dplyr
• Visualizing data with ggplot2
...Still overwhelming!

This session has a long handout with many examples.
Cleaning strings & creating summaries

Program

Import → Tidy → Transform → Communicate → Visualise → Model → Understand
Load in ebooks data

Set your working directory

```r
> setwd("C:/Users/iakovakis/Documents/ALCTS R Webinar")
```

Read in the data file

```r
> ebooks <- read.csv("./ebooks.csv"
, stringsAsFactors = F
, colClasses = c("ISBN.Print" = "character"
, "ISBN.Electronic" = "character")
)
The stringr package is essential for working with character strings.

- It is included in the tidyverse installation

Library data contains lots of messy character strings, such as

- Titles (journals, databases, books)
- Subject headings
- ISBNs/ISSNs (even though they are digits, they are treated as characters because they are identifiers)

```r
> install.packages("stringr")
> library("stringr")
```
> mean(ebooks$User.Sessions)
[1] NA
Warning message:
In mean.default(ebooks$User.Sessions) :
  argument is not numeric or logical: returning NA

Why can’t I get the mean user sessions?

> class(ebooks$User.Sessions)
[1] "character"

The vector class is not numeric or logical.
String problems

Why did R read this in as character?

> unique(ebooks$User.Sessions)
[1] "1" "2" "7" "3" "14" "15" "6" "8" "5" "12" "23" "4" "18" "42" "11" "9"
[17] "10" "29" "24" "16" "22" "34" "43" "56" "27" "146" "1,123" "123" "25" "31" "19" "13"
[33] "21" "46" "39" "285" "28" "697" "17" "240" "147" "49" "48" "109" "57" "20" "76" "30"
[49] "51" "53" "41" "26" "98" "44" "97" "115" "50" "33" "207" "61" "38" "81" "175" "134"
[65] "66" "155" "94" "99" "89" "331" "75" "182" "217" "82" "135" "52" "32" "213" "254" "36"
[81] "129" "37" "65"

THE COMMA!
In fact, the Total Pages, Pages Viewed, and Pages Printed are all read in as character due to the use of a comma in the thousands place.

```r
> str(select(ebooks, Pages.Viewed:User.Sessions))
'data.frame': 10000 obs. of 4 variables:
$ Total.Pages : chr "434" "158" "226" "116" ...
$ Pages.Viewed : chr "11" "12" "1" "2" ...
$ Pages.Copied : int 0 0 0 0 0 0 0 0 0 0 ...  
$ Pages.Printed: chr "0" "0" "0" "0" ...
$ User.Sessions: chr "1" "1" "1" "2" ...
```
`str_replace()` is similar to Find/Replace in Microsoft Excel

`help(str_replace)`
stringr::str_replace

> vec <- c("800", "900", "1,000")

> vec2 <- str_replace(vec
   , pattern = ",",
   , replacement = "")

Arguments to str_replace():
- **string**: The original vector (must be a vector, not a data frame)
- **pattern**: The pattern to look for
- **replacement**: What to replace it with (if nothing, an empty set of quotes "")
stringr::str_replace

We’ve remove the comma but still have a problem: the vector is still characters, meaning you can’t do any mathematical operations with them.

```r
> class(vec2)
[1] "character"
```

You must use `as.integer()` to coerce it to integer values.

```r
> vec2 <- as.integer(vec2)
> vec2
[1] 800 900 1000
```
On the ebooks data, we can use `mutate()` in combination with `as.integer()` and `str_replace` to replace commas in all the affected variables.

```r
ebooks <- mutate(ebooks,
  Total.Pages = as.integer(str_replace(ebooks$Total.Pages, ",", "")),
  Pages.Viewed = as.integer(str_replace(ebooks$Pages.Viewed, ",", "")),
  Pages.Printed = as.integer(str_replace(ebooks$Pages.Printed, ",", "")),
  User.Sessions = as.integer(str_replace(ebooks$User.Sessions, ",", "")))
```
stringr::str_sub

str_sub():
   takes a vector, a starting character, and an ending character, and grabs the characters in between those points

> ebooks$LC.Call[1]
[1] "M_3275.c86 1999eb"

> str_sub(ebooks$LC.Call[1]
   , start = 1
   , end = 1)
[1] "M"
Use `str_sub()` in combination with `mutate()` on the `LC.Call` variable.

This will create a new variable `LC.Class` consisting of only the first character from `LC.Call`.

```r
ebooks <- mutate(ebooks, LC.Class = str_sub(LC.Call, start = 1, end = 1))```

Hosted by ALCTS, Association for Library Collections and Technical Services
**strtrim**: removes whitespace from start and end of string

```r
> vec3 <- c(" a", "b ", " c ")

> vec3
[1] " a" "b " " c "

> str_trim(vec3)
[1] "a" "b" "c"
```
stringr::str_detect

str_detect():
Detect the presence or absence of a pattern in a string.

> vec3 <- c(" a", "b ", " c ")

> vec3
[1] " a" "b " " c "

> str_trim(vec3)
[1] "a" "b" "c"
Grouping & summarizing data with **dplyr**

**group_by()**: Cluster the dataset together in the variables you specify

**summarize()**: Use on grouped data to get summary information
`dplyr::group_by`:

Cluster the dataset together in the variables you specify.

```r
> byLC <- group_by(ebooks, LC.Class)
```

On its own, it just creates a grouped data frame.

It should be used in conjunction with `summarize()` to get summary information by group.
**summarize()**: Use on grouped data to get summary information.

Not particularly helpful without `group_by()`, because it is only one group!

```r
> summarize(ebooks, mean(User.Sessions))
mean(User.Sessions)
1 3.4309

> mean(ebooks$User.Sessions)
[1] 3.4309
```
> UserSessions_byLC <- summarize(byLC, 
  totalSessions = sum(User.Sessions))

This takes the grouped byLC dataframe and sums up the total number of User Sessions per call number.

It then returns that in the form of a data frame.
View(UserSessions_byLC)
```
dplyr::summarize
View(arrange(UserSessions_byLC, desc(totalSessions)))
```
The `dplyr` pipe `%>%`

The pipe `%>%` combines multiple `dplyr` operations

This helps so you don’t have to create several intermediate (junk) data frames like `byLC` and makes for easier to read code.
summarize, group_by and %>%

```r
> byLCSummary <- ebooks %>%
    group_by(LC.Class) %>%
    summarize(
        count = n(),
        totalSessions = sum(User.Sessions),
        avgSessions = mean(User.Sessions)) %>%
    arrange(desc(totalSessions))
```

Creating three new variables:
- **count**: use `n()` to get the number of books per call number
- **totalSessions**: sum of User Sessions per call number
- **avgSessions**: average User Sessions per call number
```r
dplyr::summarize

View(byLCSummary)

- **H** has the highest number of items and User Sessions
- **Q** has the highest average sessions, which means that these items are getting more heavily used
  - (or at least some of them are)
```

<table>
<thead>
<tr>
<th>LC.Class</th>
<th>count</th>
<th>totalSessions</th>
<th>avgSessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>2246</td>
<td>7346</td>
<td>3.270703</td>
</tr>
<tr>
<td>Q</td>
<td>1187</td>
<td>5562</td>
<td>4.685762</td>
</tr>
<tr>
<td>L</td>
<td>814</td>
<td>3248</td>
<td>3.990172</td>
</tr>
<tr>
<td>R</td>
<td>949</td>
<td>2963</td>
<td>3.122234</td>
</tr>
<tr>
<td>B</td>
<td>799</td>
<td>2929</td>
<td>3.665832</td>
</tr>
<tr>
<td>T</td>
<td>731</td>
<td>2592</td>
<td>3.545828</td>
</tr>
<tr>
<td>P</td>
<td>1067</td>
<td>2359</td>
<td>2.210872</td>
</tr>
<tr>
<td>D</td>
<td>478</td>
<td>1417</td>
<td>2.964435</td>
</tr>
<tr>
<td>G</td>
<td>206</td>
<td>1290</td>
<td>4.358108</td>
</tr>
<tr>
<td>E</td>
<td>344</td>
<td>1142</td>
<td>3.319787</td>
</tr>
<tr>
<td>K</td>
<td>221</td>
<td>885</td>
<td>4.004525</td>
</tr>
<tr>
<td>F</td>
<td>191</td>
<td>837</td>
<td>4.382199</td>
</tr>
<tr>
<td>J</td>
<td>237</td>
<td>742</td>
<td>3.130802</td>
</tr>
<tr>
<td>N</td>
<td>125</td>
<td>290</td>
<td>2.320000</td>
</tr>
<tr>
<td>Z</td>
<td>99</td>
<td>259</td>
<td>2.616162</td>
</tr>
<tr>
<td>S</td>
<td>65</td>
<td>151</td>
<td>2.323077</td>
</tr>
<tr>
<td>M</td>
<td>61</td>
<td>110</td>
<td>1.803279</td>
</tr>
<tr>
<td>U</td>
<td>40</td>
<td>78</td>
<td>1.950000</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>49</td>
<td>1.633333</td>
</tr>
<tr>
<td>A</td>
<td>12</td>
<td>48</td>
<td>4.000000</td>
</tr>
<tr>
<td>NA</td>
<td>4</td>
<td>7</td>
<td>1.750000</td>
</tr>
<tr>
<td>V</td>
<td>4</td>
<td>5</td>
<td>1.250000</td>
</tr>
</tbody>
</table>

Hosted by ALCTS, Association for Library Collections and Technical Services
If you’re a statistics person, you have many other options:

• `median()`
• `sd()`: standard deviation
• `IQR()`: interquartile range
• `min()` and `max()`
• `quantile()`
• `distinct_n()`: number of unique values
• `var()`: variance
Summary statistics

```r
> byLCSummary <- ebooks %>%
  group_by(LC.Class) %>%
  summarize(
    count = n(),
    totalSessions = sum(User.Sessions),
    avgSessions = mean(User.Sessions),
    medianSessions = median(User.Sessions),
    sdSessions = sd(User.Sessions),
    highestSession = max(User.Sessions),
    lowestSession = min(User.Sessions),
    varianceSession = var(User.Sessions) %>%
    arrange(desc(sdSessions))
```
<table>
<thead>
<tr>
<th>LC.Class</th>
<th>count</th>
<th>totalSessions</th>
<th>avgSessions</th>
<th>medianSessions</th>
<th>sdSessions</th>
<th>highestSession</th>
<th>lowestSession</th>
<th>varianceSessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>1187</td>
<td>5562</td>
<td>4.665762</td>
<td>1.0</td>
<td>34.0974176</td>
<td>1123</td>
<td>1</td>
<td>1162.6338655</td>
</tr>
<tr>
<td>B</td>
<td>799</td>
<td>2929</td>
<td>3.665832</td>
<td>1.0</td>
<td>25.0992460</td>
<td>697</td>
<td>1</td>
<td>629.9721519</td>
</tr>
<tr>
<td>G</td>
<td>296</td>
<td>1290</td>
<td>4.358108</td>
<td>1.0</td>
<td>22.0660058</td>
<td>285</td>
<td>1</td>
<td>486.9086120</td>
</tr>
<tr>
<td>K</td>
<td>221</td>
<td>885</td>
<td>4.004525</td>
<td>1.0</td>
<td>16.1037821</td>
<td>213</td>
<td>1</td>
<td>259.3317976</td>
</tr>
<tr>
<td>L</td>
<td>814</td>
<td>3248</td>
<td>3.990172</td>
<td>1.0</td>
<td>14.9231702</td>
<td>254</td>
<td>1</td>
<td>222.7010103</td>
</tr>
<tr>
<td>F</td>
<td>191</td>
<td>837</td>
<td>4.382199</td>
<td>1.0</td>
<td>12.3115477</td>
<td>146</td>
<td>1</td>
<td>151.5742078</td>
</tr>
<tr>
<td>H</td>
<td>2246</td>
<td>7346</td>
<td>3.270703</td>
<td>1.0</td>
<td>9.9546726</td>
<td>331</td>
<td>1</td>
<td>99.0955066</td>
</tr>
<tr>
<td>J</td>
<td>237</td>
<td>742</td>
<td>3.130802</td>
<td>1.0</td>
<td>9.1255242</td>
<td>129</td>
<td>1</td>
<td>83.2751913</td>
</tr>
<tr>
<td>T</td>
<td>731</td>
<td>2592</td>
<td>3.545828</td>
<td>1.0</td>
<td>8.0295017</td>
<td>123</td>
<td>1</td>
<td>54.4728970</td>
</tr>
<tr>
<td>R</td>
<td>649</td>
<td>2063</td>
<td>3.122234</td>
<td>1.0</td>
<td>6.4184121</td>
<td>134</td>
<td>1</td>
<td>41.1900136</td>
</tr>
<tr>
<td>D</td>
<td>478</td>
<td>1417</td>
<td>2.964435</td>
<td>1.0</td>
<td>5.4021495</td>
<td>81</td>
<td>1</td>
<td>29.1832189</td>
</tr>
<tr>
<td>E</td>
<td>344</td>
<td>1142</td>
<td>3.319767</td>
<td>1.0</td>
<td>4.9471921</td>
<td>41</td>
<td>1</td>
<td>24.4747101</td>
</tr>
<tr>
<td>S</td>
<td>65</td>
<td>151</td>
<td>2.323077</td>
<td>1.0</td>
<td>4.8801501</td>
<td>31</td>
<td>1</td>
<td>23.8158564</td>
</tr>
<tr>
<td>N</td>
<td>125</td>
<td>290</td>
<td>2.320000</td>
<td>1.0</td>
<td>3.3687367</td>
<td>33</td>
<td>1</td>
<td>11.3483871</td>
</tr>
<tr>
<td>A</td>
<td>12</td>
<td>48</td>
<td>4.000000</td>
<td>3.5</td>
<td>3.3574882</td>
<td>13</td>
<td>1</td>
<td>11.2727273</td>
</tr>
<tr>
<td>Z</td>
<td>99</td>
<td>259</td>
<td>2.615162</td>
<td>1.0</td>
<td>2.7206957</td>
<td>19</td>
<td>1</td>
<td>7.4021851</td>
</tr>
<tr>
<td>P</td>
<td>1067</td>
<td>2359</td>
<td>2.210872</td>
<td>1.0</td>
<td>2.6470901</td>
<td>29</td>
<td>1</td>
<td>7.0070652</td>
</tr>
<tr>
<td>M</td>
<td>61</td>
<td>110</td>
<td>1.808279</td>
<td>1.0</td>
<td>2.2122852</td>
<td>13</td>
<td>1</td>
<td>4.8939891</td>
</tr>
<tr>
<td>U</td>
<td>40</td>
<td>78</td>
<td>1.950000</td>
<td>1.0</td>
<td>1.9605075</td>
<td>11</td>
<td>1</td>
<td>3.8435897</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>49</td>
<td>1.633333</td>
<td>1.0</td>
<td>1.7710952</td>
<td>10</td>
<td>1</td>
<td>3.1367816</td>
</tr>
<tr>
<td>NA</td>
<td>4</td>
<td>7</td>
<td>1.750000</td>
<td>1.5</td>
<td>0.9574271</td>
<td>3</td>
<td>1</td>
<td>0.9166667</td>
</tr>
<tr>
<td>V</td>
<td>4</td>
<td>5</td>
<td>1.250000</td>
<td>1.0</td>
<td>0.5000000</td>
<td>2</td>
<td>1</td>
<td>0.2500000</td>
</tr>
</tbody>
</table>

Hosted by ALCTS, Association for Library Collections and Technical Services
Mind the P and Q

PQSummary <- filter(byLCSummary, LC.Class == "P" | LC.Class == "Q")

<table>
<thead>
<tr>
<th>LC.Class</th>
<th>count</th>
<th>totalSessions</th>
<th>avgSessions</th>
<th>medianSessions</th>
<th>sdSessions</th>
<th>highestSession</th>
<th>lowestSession</th>
<th>varianceSessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>1187</td>
<td>5562</td>
<td>4.685762</td>
<td>1</td>
<td>34.09742</td>
<td>1123</td>
<td>1</td>
<td>1162.633886</td>
</tr>
<tr>
<td>P</td>
<td>1067</td>
<td>2359</td>
<td>2.210872</td>
<td>1</td>
<td>2.64709</td>
<td>29</td>
<td>1</td>
<td>7.007086</td>
</tr>
</tbody>
</table>

Despite having comparable numbers of items (1187 and 1067), Q has a much higher standard deviation and variance.

Usage of P books is lower and much more evenly distributed.

Q has at least one serious outlier.
Visualizing data with ggplot2
Visualizing data

• Base R includes several functions for quick data visualization
  • plot()
  • barplot()
  • hist()
  • boxplot()

• However, most R users consider ggplot to be the most elegant and versatile
ggplot2

• As dplyr implements a “grammar” for data manipulation, ggplot2 does so for data visualization
• Also created by Hadley Wickham
• As always, start by loading the package

> library(ggplot2)

Portions of this section are adapted from the Data Visualization chapter of R for Data Science: http://r4ds.had.co.nz/data-visualisation.html
• Will plot directly to the **Plots** tab in the Navigation Pane (lower right)
• But you also assign it to a variable and call `print()`
• `ggplot()` creates the initial coordinate system (a “blank canvas”) that you then add layers to
• Functions are added with `+`

```r
> ggplot(data = ebooks)
```
Data is visualized in the canvas with geometric shapes, what are called geom functions.

- ggplot2 contains over 30 geoms. For example,
  - Bar plots use geom_bar()
  - Histograms use geom_histogram()
  - Line plots use geom_line()
  - Scatter plots use geom_point()
Add a layer of bars to the plot with `geom_bar()`. This creates a bar plot:

```r
> ggplot(data = ebooks) +
  geom_bar(mapping = aes(x = LC.Class))
```
aes: Mapping aesthetic

- Each `geom` takes a `mapping` argument
- This is paired with `aes()`, the `aesthetic mapping` argument
- The `x` and `y` arguments to `aes` specify which variables to map to the `x` and `y` axes
- `ggplot` looks for the mapped variable in the `data` argument
In this case, the LC.Class is being mapped to the x axis

```r
> ggplot(data = ebooks) +
  geom_bar(aes(x = LC.Class))
```

Note that the `count` is not a variable in the original dataset.

`geom_bar` is **binning** your data and plotting the bin counts: the number of items falling into each bin.

You don’t need to specify `mapping =` and it will henceforth be omitted.
Univariate: one variable

```r
> ggplot(data = ebooks, aes(x = User.Sessions)) + geom_histogram(`stat_bin()` using `bins = 30`). Pick better value with `binwidth`.
```

**Problem:** The overwhelming majority of books have a small amount of usage. This throws off the scale.
Changing scales

I thought about using different data, but this is actually a common problem in usage analysis: The vast majority of items have little to no usage.

Also, this illustrates a way to get around the problem in `ggplot`: change the x and y scales
> ggplot(data = ebooks) +
  geom_histogram(aes(x = User.Sessions), binwidth = .5) +
  scale_x_log10() +
  scale_y_log10()

Notice the scale goes from 10 to 100 to 1,000.

So over 1,000 ebooks have 1-5 user sessions,

And a very small number have over 1,000 user sessions.
```r
> ggplot(data = ebooks) +
  geom_histogram(aes(x = User.Sessions), binwidth = .5) +
  scale_x_log10() +
  scale_y_log10()
```
Let’s take a look at some higher usage items.

First, create a filtered data set with extreme outliers removed:

```r
> ebooksPlot <- filter(ebooks,
                      User.Sessions < 500 & User.Sessions > 10)
```
Bivariate geoms

Plot user sessions by call number class.

Still using a logarithmic scale

```r
> ggplot(data = ebooksPlot) +
  geom_point(aes(x = LC.Class, y = User.Sessions)) +
  scale_y_log10()
```
Again, notice the scale.

A few quick observations:

- P has no items with over 10 user sessions
- G has very few items, but 3 of them have over 100 sessions
Bivariate

`geom_jitter()`
Same as `geom_point` but adds a bit of vertical space between dots
ggplot(data = ebooksPlot) + geom_point(aes(x = LC.Class, y = User.Sessions)) + scale_y_log10()
Add a third variable with aes()

We previously mapped variables to the x and y axes using aes().

We will now map a third variable to a visual aesthetic, color

```r
> ggplot(data = ebooksPlot) +
  geom_point(aes(x = LC.Class,
                 y = User.Sessions,
                 color = Collection)) +
  scale_y_log10()
```

Associate the name of the aesthetic (color) to the name of the variable (Collection) inside aes()
**aes(color)**

`ggplot` automatically assigns a unique level of the aesthetic to each unique value of the variable (this is called **scaling**).

We can see that most high usage titles are subscription or librarian-purchased rather than DDA.
Use fill with `geom_bar` to create a stacked bar plot

```r
> ggplot(data = ebooks) + geom_bar(aes(x = LC.Class, fill = Collection))
```
Add a third variable with facets

The **facet** functions create subplots of subsets of the data

```r
> ggplot(data = ebooksPlot) +
  geom_point(aes(x = User.Sessions,
                  y = Pages.Viewed)) +
  `facet_wrap(~ LC.Class)` +
  geom_smooth(aes(x = User.Sessions,
                  y = Pages.Viewed)) +
  scale_x_log10()
```
Use with `%>%`

```r
> filter(ebooks
  , LC.Class == "Q" | LC.Class == "P") %>%
  ggplot() +
  geom_freqpoly(aes(x = Pages.Viewed
  , color = LC.Class
  , binwidth = .5)
  , size = 1.5) +
  scale_x_log10()
```
Style the plot with theme()

You have almost total control over the appearance of your plot.

There are a number of preconstructed themes, which you can view at the tidyverse ggtheme page:
http://ggplot2.tidyverse.org/reference/ggtheme.html

You can call help(theme) to view the multiple parameters to add to the theme call to construct your own custom theme.
Save the plot with `ggsave()`

Call `help(ggsave)` to read the various arguments you can pass to save your plot, include the filename, path, size, file type (`device`), dimensions, and dpi

```r
> ggsave(filename = "myPlot.png",
        plot = myPlot,
        path = ".:/doc/images/",
        device = "png",
        height = 4,
        width = 4,
        units = "in")
```
R Studio shortcuts to know

Windows/Mac key

- Run current line/selection in script pane: Ctrl/Cmd + Enter
- Skip words (works in Word too): Ctrl/option + ← or →
- Scan through previously entered commands: ↑ or ↓
- Insert <-: Alt/Option + -
- Jump from Console to Script pane: Ctrl + 1
- Jump from Script pane to Console: Ctrl + 2
- Fix indentation in script pane: Highlight text and press Ctrl + i
- Rerun previous chunk of code: Ctrl/Cmd + Shift + P

Hosted by ALCTS, Association for Library Collections and Technical Services
Victory!

Continue working through the handouts and help pages. If there is enough continuing interest, perhaps we can form an R (support) Group.
Questions?