Subsection 1

tidyr

Additionally Complex Analysis

- Often times, the data we are trying to plot simply is not in the format we need it to be
- It is not uncommon that a majority of the time spent visualizing data is not writing the visualization code, e..g, `ggplot`, but rather restructuring data (mainly data frames) so as to be able to visualize the data
- There are functions in base R such as `aggregate()` and `merge()`, etc., which can help with this, as well as functions such as `melt()` and `cast()` from the `reshape2` package
- This section will examine the use of `dplyr`

Tidy Data with tidyr

- Data should always be tidy before you begin to work with it
- All data should be organized such that
  1. **each column is a variable**
  2. **each row is an observation**
- tidyr provides four main functions for tidying your messy data
  1. gather()
  2. spread()
  3. separate()
  4. unite()

tidyr::gather()

- gather() takes multiple columns and gathers them into key-value pairs
- It makes *wide* data *longer*
- The equivalent function in the popular reshape2 package is melt()

tidyr::gather() [EXAMPLE]

- A company wants to compare sales of customers who have member cards and those who don't in five different cities

```
> (sales <- read_csv("tidyr_gather_memberCardSales.csv"))
Parsed with column specification:
cols(
  city = col_character(),
  memberCardSales = col_integer(),
  noMemberCardSales = col_integer()
)
# A tibble: 5  3
          city memberCardSales noMemberCardSales
         <chr>           <int>             <int>
1      New York          297783            756185
2 San Francisco          761946            485681
3       Chicago          769215            155030
4        Austin          604328            962551
5     Las Vegas          174663            420949
```

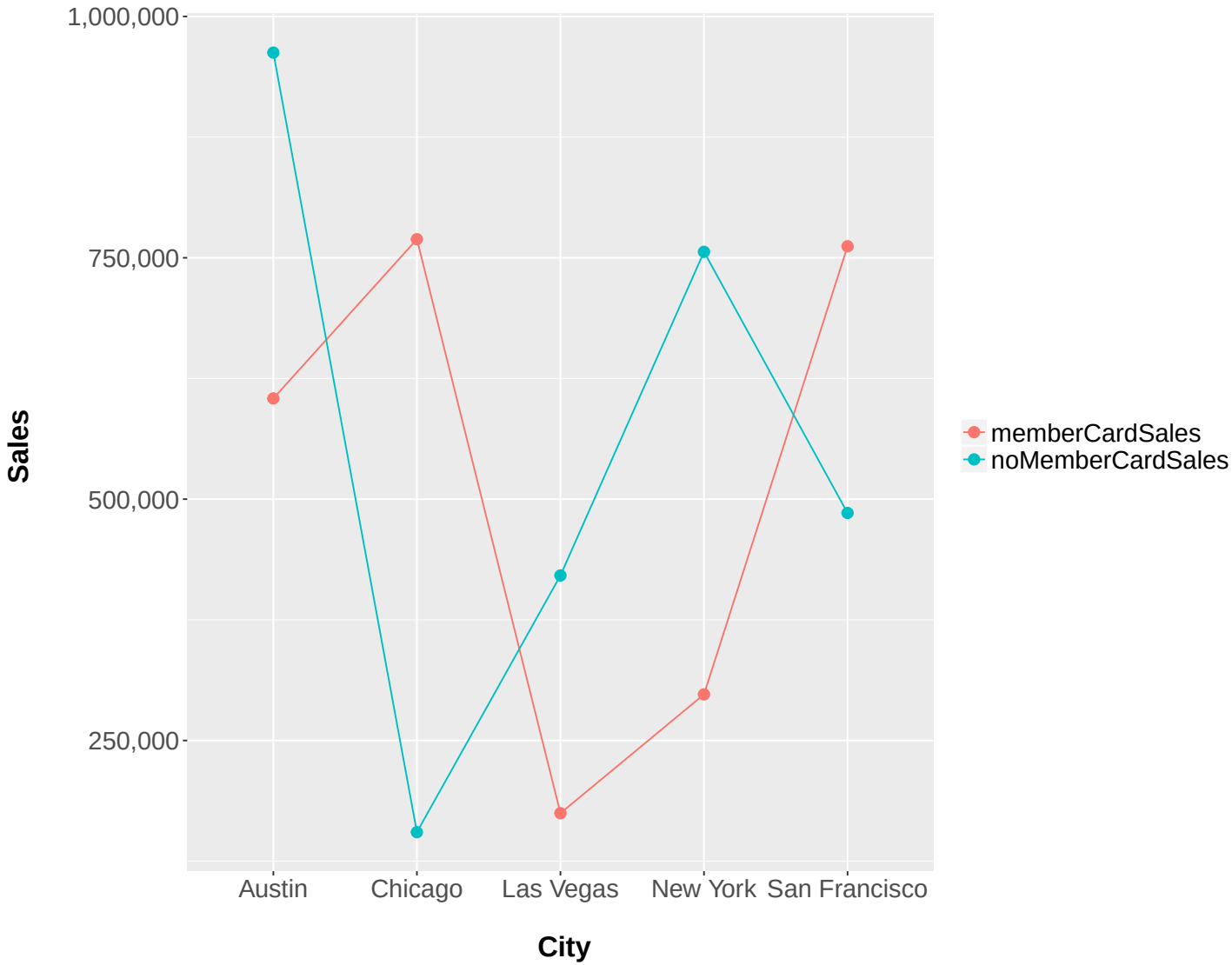
tidyr::gather() [EXAMPLE]

- A company wants to compare sales of customers who have member cards and those who don't in five different cities

```
> (sales <- read_csv("tidyr_gather_memberCardSales.csv"))
Parsed with column specification:
cols(
  city = col_character(),
  memberCardSales = col_integer(),
  noMemberCardSales = col_integer()
)
# A tibble: 5  3
           city memberCardSales noMemberCardSales
          <chr>           <int>              <int>
1      New York          297783             756185
2 San Francisco          761946             485681
3       Chicago          769215             155030
4        Austin          604328             962551
5     Las Vegas          174663             420949
```

- OBJECTIVE Create a line plot in ggplot of Sales (y) on City (x), with one line for card status

tidyr::gather() [EXAMPLE CONT'D]



tidyr::gather() [EXAMPLE CONT'D]

- Recall that in a tidy data set, each column is a variable and each row is an observation, which is not the case here
- Use `gather()` to organize the columns into key-value pairs of `cardStatus` and `Sales`

```
> (myTidySales <- sales %>% gather(cardStatus,
    Sales, memberCardSales, noMemberCardSales))

# A tibble: 10 x 3
            city        cardStatus  Sales
           <chr>             <chr>  <int>
1       New York   memberCardSales 297783
2  San Francisco   memberCardSales 761946
3        Chicago   memberCardSales 769215
4         Austin   memberCardSales 604328
5      Las Vegas   memberCardSales 174663
6       New York noMemberCardSales 756185
7  San Francisco noMemberCardSales 485681
8        Chicago noMemberCardSales 155030
9         Austin noMemberCardSales 962551
10     Las Vegas noMemberCardSales 420949
```

tidyr::gather() [GGPLOT CODE]

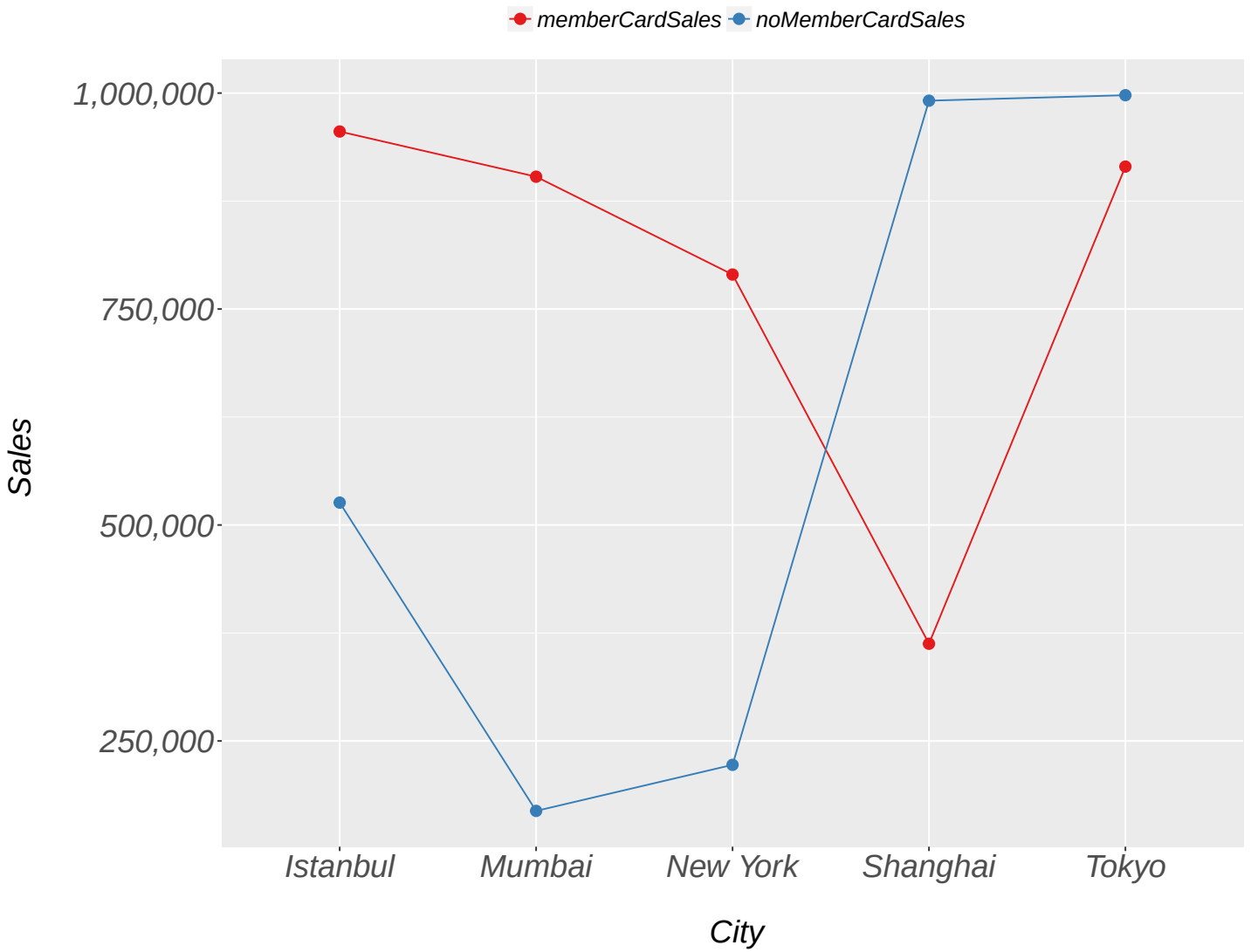
```
library(scales)

ggplot(myTidySales,
  aes(x = city, y = Sales, group = cardStatus, colour = cardStatus)) +
  geom_line() +
  geom_point(size = 3) +
  scale_y_continuous(labels = comma) +
  ylab("Sales \n") +
  xlab("\n City") +
  theme(legend.title = element_blank(),
        axis.text = element_text(size = 16),
        axis.title = element_text(size = 18, face = "bold"),
        legend.text = element_text(size = 16))
```

tidyr::separate()

- Sometimes variables are clumped together in a single column
- `separate()` allows you to parse them
- `extract()` works similarly but uses regexp groups instead of a splitting pattern or position

- EXAMPLE A company wants to compare sales of customers from *five international cities* who have member cards and those who don't
- OBJECTIVE Create a line plot in `ggplot` of Sales (y) on City (x), with one line for card status

tidyr::separate() [EXAMPLE CONT'D]



tidyr::separate() [EXAMPLE]

```
> (sales <- read_csv("tidyr_separate_memberCardSales.csv"))
Parsed with column specification:
cols(
  city = col_character(),
  noMemberCardSales = col_character(),
  memberCardSales = col_character()
)
# A tibble: 5  3
      city noMemberCardSales memberCardSales
     <chr>             <chr>           <chr>
1 New York         222355USD       789924USD
2    Tokyo         997512JPY       914879JPY
3   Mumbai         169019INR       903189INR
4 Istanbul         525930TRY       955508TRY
5 Shanghai         991163CNY       362563CNY
```

- The first challenge is how to create a numeric value out of a currency column that contains alphanumeric characters

tidyr::separate() [EXAMPLE CONT'D]

- Firstly, observe from the raw data (previous slide) that it is not tidy, i.e., each column should be a variable and each row should be an observation
- STEP 1 Make the data tidy using gather()

```
> (myTidySales <- sales %>% gather(cardStatus, Sales,
    memberCardSales, noMemberCardSales))

# A tibble: 10  3
       city        cardStatus     Sales
      <chr>             <chr>     <chr>
1  New York   memberCardSales 789924USD
2     Tokyo   memberCardSales 914879JPY
3    Mumbai   memberCardSales 903189INR
4  Istanbul   memberCardSales 955508TRY
5  Shanghai   memberCardSales 362563CNY
6  New York noMemberCardSales 222355USD
7     Tokyo noMemberCardSales 997512JPY
8    Mumbai noMemberCardSales 169019INR
9  Istanbul noMemberCardSales 525930TRY
10 Shanghai noMemberCardSales 991163CNY
```

tidyr::separate() [EXAMPLE CONT'D]

- STEP 2 Split Sales into sales and currency w/ separate()

```
> (myTidierDataFrame <- myTidySales %>% separate(Sales,
    into = c("sales", "currency"), sep = -4, convert = T))
# A tibble: 10  4
       city        cardStatus  sales currency
*     <chr>             <chr>  <int>    <chr>
1  New York   memberCardSales 789924      USD
2     Tokyo   memberCardSales 914879      JPY
3    Mumbai   memberCardSales 903189      INR
4  Istanbul   memberCardSales 955508      TRY
5  Shanghai   memberCardSales 362563      CNY
6  New York noMemberCardSales 222355      USD
7     Tokyo noMemberCardSales 997512      JPY
8    Mumbai noMemberCardSales 169019      INR
9  Istanbul noMemberCardSales 525930      TRY
10 Shanghai noMemberCardSales 991163      CNY
```

- The sep = -4 option counts from the far right -3 spaces (a quirk)
- The convert = T option tells separate() to convert the new columns into types it thinks is most appropriate

tidyr::separate() [GGPLOT CODE]

```
library(scales)

ggplot(myTidierDataFrame, aes(x = city, y = sales)) +
  geom_line(aes(group = cardStatus, colour = cardStatus)) +
  geom_point(aes(group = cardStatus, colour = cardStatus), size = 3) +
  labs(x = "\n City", y = "Sales \n") +
  scale_y_continuous(labels = comma) +
  theme(legend.title = element_blank(),
        axis.title = element_text(size = 20,face = "italic"),
        axis.text = element_text(size = 20, face = "italic"),
        legend.text = element_text(size = 14,face = "italic"),
        legend.position = "top") +
  scale_colour_brewer(palette = "Set1")
```

tidyr::spread() & tidyr::unite()

- The opposite of the `gather()` function is the `spread()` function, which takes *long* data and makes it *wide*
- The opposite of the `separate()` function is the `unite()` function, which combines multiple columns into a single column