Wide Versus Narrow Format

```
# As usual, we load the mosaic and tidyverse packages.
# Loading tidyverse automatically loads the dplyr package.
# We want to look at two operations from dplyr: pivot_wider()
# and pivot_longer().
library(mosaic)
library(tidyverse)

# Let's work with the babynames data frame from the package with the
# same name.
library(babynames)
head(babynames)
```

```
## # A tibble: 6 × 5
##    year sex   name          n   prop
##   <dbl> <chr> <chr>     <int>  <dbl>
## 1  1880 F     Mary       7065 0.0724
## 2  1880 F     Anna       2604 0.0267
## 3  1880 F     Emma       2003 0.0205
## 4  1880 F     Elizabeth  1939 0.0199
## 5  1880 F     Minnie     1746 0.0179
## 6  1880 F     Margaret   1578 0.0162
```

```
# Let's see what the maximum value of the year variable is.
max(babynames$year)
```

```
## [1] 2017
```

```
# We are interested in names that are often used for both male
# and female babies.  I chose ten such names; I encourage you to
# try your own favorites!
gender_neutral_names <- c("Adrian","Blake","Charlie",
                          "Drew","Jordan","Kyle",
                          "Morgan","Parker","Quinn","Taylor")
gender_neutral_names
```

```
##  [1] "Adrian"  "Blake"   "Charlie" "Drew"    "Jordan"  "Kyle"    "Morgan"
##  [8] "Parker"  "Quinn"   "Taylor"
```

```
# Let's filter babynames for these names and the years from
# 2010 to 2017.  Also, we don't need the prop column, we
# rename the sex and n columns, and change the types of the
# year and sex columns.
data01 <- babynames %>%
  filter(name %in% gender_neutral_names & year > 2009) %>%
  select(!prop) %>%
  rename(gender = sex, count = n) %>%
```

```
  mutate(year = as.character(year), gender = as.factor(gender))
head(data01,20)
```

```
## # A tibble: 20 × 4
##    year  gender name    count
##    <chr> <fct>  <chr>   <int>
##  1 2010  F      Taylor   5894
##  2 2010  F      Morgan   4072
##  3 2010  F      Jordan   1726
##  4 2010  F      Quinn    1278
##  5 2010  F      Charlie   670
##  6 2010  F      Parker    652
##  7 2010  F      Blake     241
##  8 2010  F      Drew      167
##  9 2010  F      Adrian    163
## 10 2010  F      Kyle       33
## 11 2010  M      Jordan   8230
## 12 2010  M      Adrian   7405
## 13 2010  M      Parker   4730
## 14 2010  M      Blake    4697
## 15 2010  M      Kyle     3572
## 16 2010  M      Charlie  1429
## 17 2010  M      Drew     1413
## 18 2010  M      Quinn    1237
## 19 2010  M      Taylor    955
## 20 2010  M      Morgan    498
```

```
# Next, we compute, for each name and gender, the total number
# of babies with that name and gender.
names_by_gender <- data01 %>%
  group_by(name,gender) %>%
  summarize(total = sum(count))
```

```
## `summarise()` has grouped output by 'name'. You can override using the
## `.groups` argument.
```

```
names_by_gender
```

```
## # A tibble: 20 × 3
## # Groups:   name [10]
##    name    gender total
##    <chr>   <fct>  <int>
##  1 Adrian  F        995
##  2 Adrian  M      54258
##  3 Blake   F       5033
##  4 Blake   M      37024
##  5 Charlie F      10549
##  6 Charlie M      12880
##  7 Drew    F       1639
##  8 Drew    M       8479
##  9 Jordan  F      10540
```

```
## 10 Jordan  M      55165
## 11 Kyle    F        415
## 12 Kyle    M      20989
## 13 Morgan  F      24174
## 14 Morgan  M       3333
## 15 Parker  F       9551
## 16 Parker  M      41204
## 17 Quinn   F      20584
## 18 Quinn   M       7993
## 19 Taylor  F      33596
## 20 Taylor  M       6127

# Goal: Determine which name is the most "gender neutral", in
# that the proportion of females with that name comes closest to
# 50%.
#
# The above data frame is said to be in "long" format, because there
# is a row for each name/gender combination.
# To accomplish our goal, we would like to get the data in
# "wide" format - that is, we would like to have just one row for
# each name, with columns name, F (number of females with that name),
# and M (number of males with that name).

# The operation to convert a data frame from long to narrow
# format is pivot_wider().
names_by_gender_wide <- names_by_gender %>%
  pivot_wider(names_from = gender, values_from = total)
names_by_gender_wide

## # A tibble: 10 × 3
## # Groups:   name [10]
##    name        F     M
##    <chr>   <int> <int>
##  1 Adrian    995 54258
##  2 Blake    5033 37024
##  3 Charlie 10549 12880
##  4 Drew     1639  8479
##  5 Jordan  10540 55165
##  6 Kyle      415 20989
##  7 Morgan  24174  3333
##  8 Parker   9551 41204
##  9 Quinn   20584  7993
## 10 Taylor  33596  6127

# To convert from wide to long format, we use the
# pivot_longer() command.
names_by_gender_narrow <- names_by_gender_wide %>%
  pivot_longer(!name, names_to = "gender", values_to = "total")
names_by_gender_narrow
```

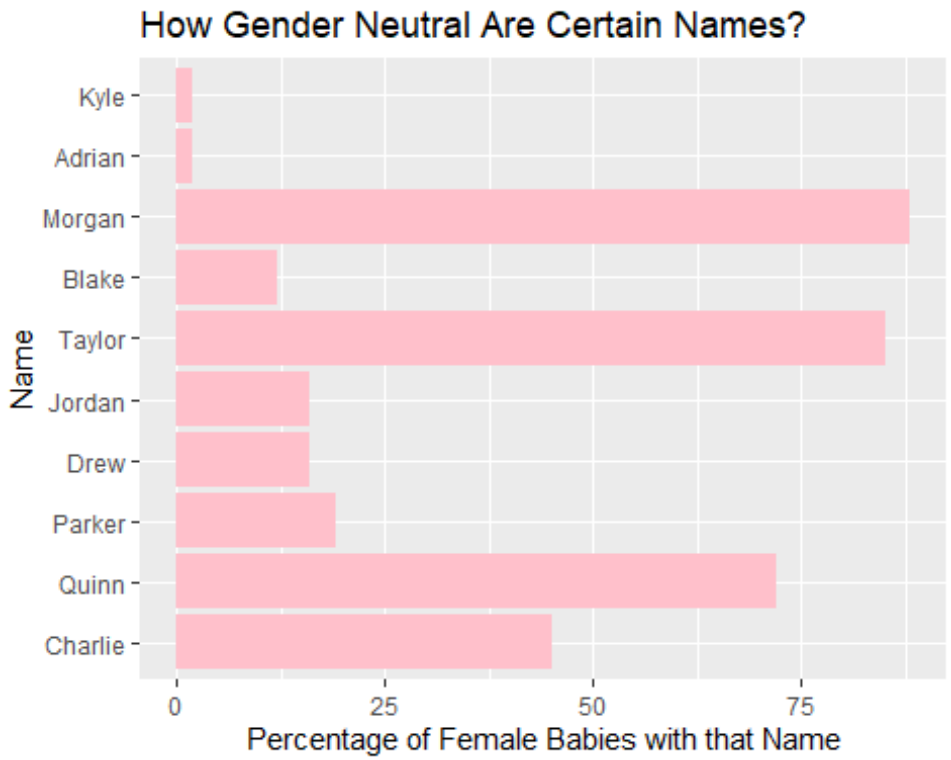
```
## # A tibble: 20 × 3
## # Groups:   name [10]
##    name    gender total
##    <chr>   <chr>  <int>
##  1 Adrian  F        995
##  2 Adrian  M      54258
##  3 Blake   F       5033
##  4 Blake   M      37024
##  5 Charlie F      10549
##  6 Charlie M      12880
##  7 Drew    F       1639
##  8 Drew    M       8479
##  9 Jordan  F      10540
## 10 Jordan  M      55165
## 11 Kyle    F        415
## 12 Kyle    M      20989
## 13 Morgan  F      24174
## 14 Morgan  M       3333
## 15 Parker  F       9551
## 16 Parker  M      41204
## 17 Quinn   F      20584
## 18 Quinn   M       7993
## 19 Taylor  F      33596
## 20 Taylor  M       6127

# For more information, use:
# vignette("pivot")

# Completing our task ...
data02 <- names_by_gender_wide %>%
  mutate(percent_F = 100*(round(F/(F + M),2)),
         diff_from_50 = abs(percent_F - 50)) %>%
  arrange(diff_from_50)
data02

## # A tibble: 10 × 5
## # Groups:   name [10]
##    name        F     M percent_F diff_from_50
##    <chr>   <int> <int>     <dbl>        <dbl>
##  1 Charlie 10549 12880        45            5
##  2 Quinn   20584  7993        72           22
##  3 Parker   9551 41204        19           31
##  4 Drew     1639  8479        16           34
##  5 Jordan  10540 55165        16           34
##  6 Taylor  33596  6127        85           35
##  7 Blake    5033 37024        12           38
##  8 Morgan  24174  3333        88           38
##  9 Adrian    995 54258         2           48
## 10 Kyle      415 20989         2           48
```

```r
# For fun, let's produce a column graph.
ggplot(data = data02,
       aes(x = percent_F, y = reorder(name,diff_from_50))) +
  geom_col(fill = "pink") +
  labs(x = "Percentage of Female Babies with that Name",
       y = "Name",
       title = "How Gender Neutral Are Certain Names?")
```




```r
# Another Example: For this example, we use the Marriage data
# frame.
# help("Marriage")
head(Marriage)
```

```
##   bookpageID    appdate ceremonydate delay     officialTitle person
dob
## 1   B230p539 1996-10-29   1996-11-09    11    CIRCUIT JUDGE   Groom 2064-
04-11
## 2   B230p677 1996-11-12   1996-11-12     0 MARRIAGE OFFICIAL  Groom 2064-
08-06
## 3   B230p766 1996-11-19   1996-11-27     8 MARRIAGE OFFICIAL  Groom 2062-
02-20
## 4   B230p892 1996-12-02   1996-12-07     5          MINISTER  Groom 2056-
05-20
## 5   B230p994 1996-12-09   1996-12-14     5          MINISTER  Groom 2066-
12-14
## 6  B230p1209 1996-12-26   1996-12-26     0 MARRIAGE OFFICIAL  Groom 1970-
02-21
```

```
##        age     race prevcount prevconc hs college dayOfBirth        sign
## 1 32.60274    White         0     <NA> 12       7        102       Aries
## 2 32.29041    White         1  Divorce 12       0        219         Leo
## 3 34.79178 Hispanic         1  Divorce 12       3         51      Pisces
## 4 40.57808    Black         1  Divorce 12       4        141      Gemini
## 5 30.02192    White         0     <NA> 12       0        348 Saggitarius
## 6 26.86301    White         1     <NA> 12       0         52      Pisces
```

```
# Note that there is a problem with the dob values!  We'll fix
# this in the tutorial on the lubridate package.
# Goal: Explore the distribution of the difference in age
# between grooms and their brides.
# We begin by selecting the columns of interest, and arranging
# the rows in order of bookpageID.
marriage_data_long <- Marriage %>%
  select(bookpageID, person, age) %>%
  mutate(bookpageID = as.character(bookpageID)) %>%
  arrange(bookpageID)
head(marriage_data_long)
```

```
##   bookpageID person      age
## 1  B230p1209  Groom 26.86301
## 2  B230p1209  Bride 25.12055
## 3  B230p1354  Groom 25.30685
## 4  B230p1354  Bride 25.14795
## 5  B230p1665  Groom 35.05205
## 6  B230p1665  Bride 20.44658
```

```
# As its name implies, the above data frame is in long format;
# For each marriage ceremony, there are two rows, one for the
# groom and one for the bride. In order to accomplish our goal,
# we need to convert to wide format.
marriage_data_wide <- marriage_data_long %>%
  pivot_wider(names_from = person, values_from = age)
head(marriage_data_wide)
```

```
## # A tibble: 6 × 3
##   bookpageID Groom Bride
##   <chr>      <dbl> <dbl>
## 1 B230p1209   26.9  25.1
## 2 B230p1354   25.3  25.1
## 3 B230p1665   35.1  20.4
## 4 B230p1948   21.3  20.0
## 5 B230p539    32.6  28.7
## 6 B230p677    32.3  52.6
```

```
# Completing our goal ...
result <- marriage_data_wide %>%
  mutate(diff_in_age = round(Groom - Bride,2))
head(result)
```

```
## # A tibble: 6 × 4
##   bookpageID Groom Bride diff_in_age
##   <chr>      <dbl> <dbl>       <dbl>
## 1 B230p1209   26.9  25.1        1.74
## 2 B230p1354   25.3  25.1        0.16
## 3 B230p1665   35.1  20.4       14.6
## 4 B230p1948   21.3  20.0        1.29
## 5 B230p539    32.6  28.7        3.88
## 6 B230p677    32.3  52.6      -20.3
```

```
ggplot(data = result,
       aes(y = diff_in_age)) +
  geom_boxplot() +
  labs(y = "Groom's Age - Bride's Age")
```