Lab 10: Tidyverse I: Pipes and Dplyr

Name: Rufus Petrie

This week's agenda: learning to master pipes and `dplyr`.

```
# Load the tidyverse!
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Pipes to base R

For each of the following code blocks, which are written with pipes, write equivalent code in base R (to do the same thing).

- **1a.**

```
# Pipes:
letters <- "abcde"
letters %>%
  toupper %>%
  paste(collapse="+")
```

```
## [1] "ABCDE"
```

```
# Base R:
paste(toupper(letters), collapse="+")
```

```
## [1] "ABCDE"
```

- **1b.**

```
# Pipes:
phrase <- "     Ceci n'est pas une pipe     "
"     Ceci n'est pas une pipe     " %>%
  gsub("une", "un", .) %>%
  trimws
```

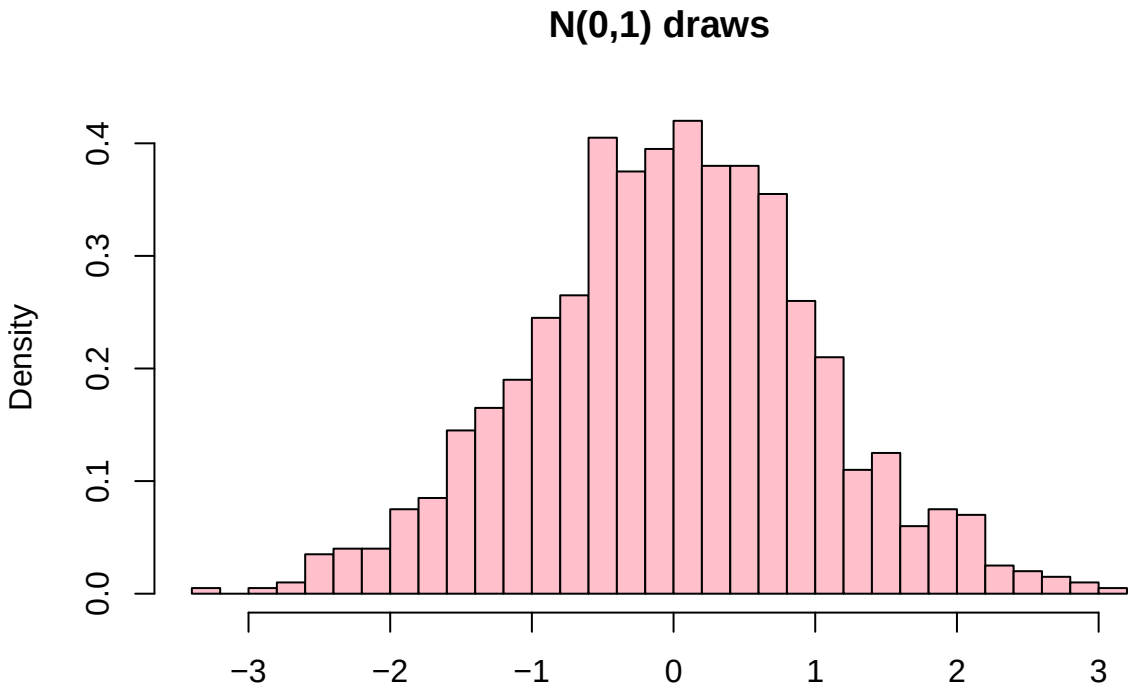
```
## [1] "Ceci n'est pas un pipe"
```

```
# Base R:
trimws(gsub("une", "un", phrase))
```

```
## [1] "Ceci n'est pas un pipe"
```

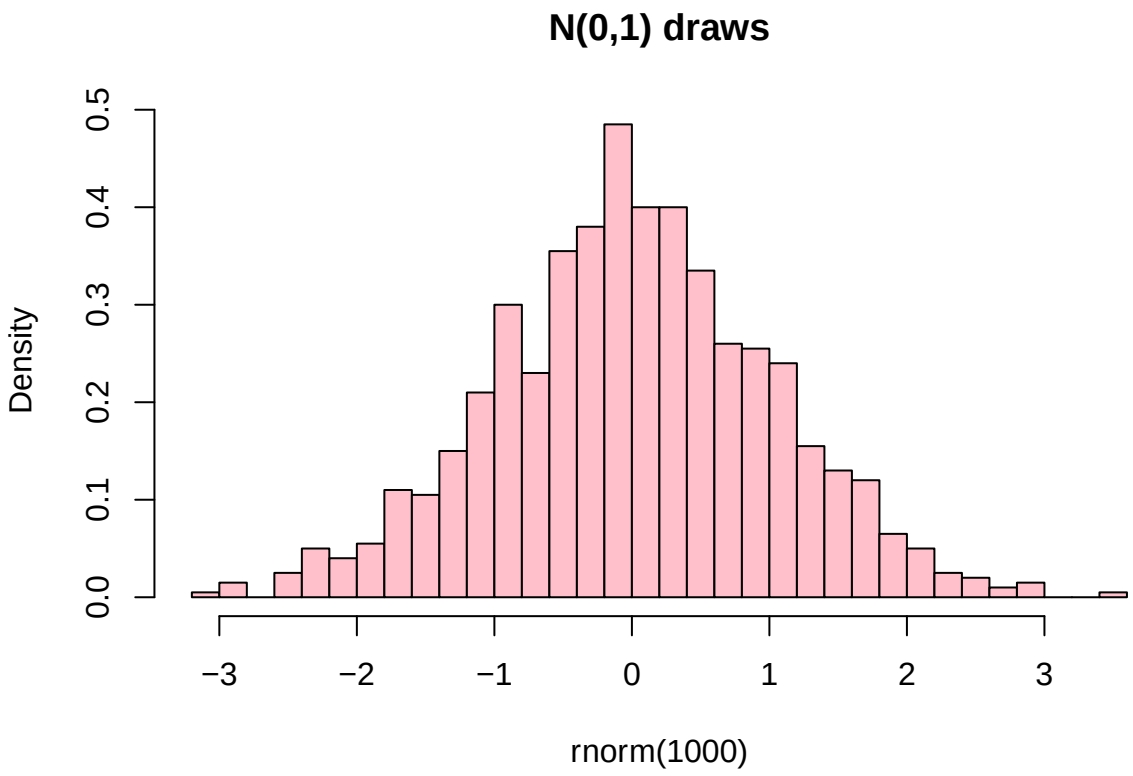
- **1c.**

```
# Pipes:
rnorm(1000) %>%
  hist(breaks=30, main="N(0,1) draws", col="pink", prob=TRUE)
```



.

```
# Base R:
hist(rnorm(1000), breaks=30, main="N(0,1) draws", col="pink", prob=TRUE)
```



- **1d.**

```
# Pipes:
rnorm(1000) %>%
  hist(breaks=30, plot=FALSE) %>%
  "[["("density") %>%
  max
```

```
## [1] 0.445
```

```
# Base R:
max(hist(rnorm(1000), breaks=30, plot=FALSE)$density)
```

```
## [1] 0.44
```

Base R to pipes

For each of the following code blocks, which are written in base R, write equivalent code with pipes (to do the same thing).

- **2a.** Hint: you'll have to use the dot `.`, as seen above in Q1b, or in the lecture notes.

```
# Base R:
paste("Your grade is", sample(c("A","B","C","D","R"), size=1))
```

```
## [1] "Your grade is A"
```

```
# Pipes:
c("A","B","C","D","R") %>%
  sample(size=1) %>%
  paste("Your grade is", .)
```

```
## [1] "Your grade is C"
```

- **2b.** Hint: you can use the dot `.` again, in order to index `state.name` directly in the last pipe command.

```
# Base R:
state.name[which.max(state.x77[,"Illiteracy"])]
```

```
## [1] "Louisiana"
```

```
# Pipes:
state.x77 %>%
  .[,"Illiteracy"] %>%
  which.max %>%
  state.name[.]
```

```
## [1] "Louisiana"
```

- **2c.** Hint: if `x` is a list of length 1, then `x[[1]]` is the same as `unlist(x)`.

```
str.url = "http://www.stat.cmu.edu/~ryantibs/statcomp/data/trump.txt"

# Base R:
lines = readLines(str.url)
text = paste(lines, collapse=" ")
words = strsplit(text, split="[[:space:]]|[[:punct:]]")[[1]]
wordtab = table(words)
wordtab = sort(wordtab, decreasing=TRUE)
head(wordtab, 10)
```

```
## words
##       the  and   of   to  our will   in    I have
##  524  189  146  127  126   90   83   69   67   58
```

```
# Pipes:
readLines(str.url) %>%
  paste(collapse=" ") %>%
  strsplit(split="[[:space:]]|[[:punct:]]") %>%
  unlist %>%
  table %>%
  sort(decreasing = TRUE) %>%
  head(10)
```

```
## .
##       the  and   of   to  our will   in    I have
##  524  189  146  127  126   90   83   69   67   58
```

- **2d.** Hint: the only difference between this and the last part is the line `words = words[words != ""]`. This is a bit tricky line to do with pipes: use the dot `.`, once more, and manipulate it as if were a variable name.

```
# Base R:
lines = readLines(str.url)
text = paste(lines, collapse=" ")
words = strsplit(text, split="[[:space:]]|[[:punct:]]")[[1]]
```

```
words = words[words != ""]
wordtab = table(words)
wordtab = sort(wordtab, decreasing=TRUE)
head(wordtab, 10)
```

```
## words
##  the  and   of   to  our will   in    I have    a
##  189  146  127  126   90   83   69   67   58   51
```

```
# Pipes:
readLines(str.url) %>%
  paste(collapse=" ") %>%
  strsplit(split="[[:space:]]|[[:punct:]]") %>%
  unlist %>%
  .[.!=""] %>%
  table %>%
  sort(decreasing = TRUE) %>%
  head(10)
```

```
## .
##  the  and   of   to  our will   in    I have    a
##  189  146  127  126   90   83   69   67   58   51
```

Sprints data, revisited

Below we read in a data frame `sprint.w.df` containing the top women's times in the 100m sprint, as seen in previous labs. We also define a function `factor.to.numeric()` that was used in Lab 8, to convert the Wind column to numeric values. In what follows, use `dplyr` and pipes to answer the following questions on `sprint.w.df`.

```
sprint.w.df = read.table(
  file="http://www.stat.cmu.edu/~ryantibs/statcomp/data/sprint.w.dat",
  sep="\t", header=TRUE, quote="", stringsAsFactors=TRUE)

factor.to.numeric = Vectorize(function(x) {
  x = strsplit(as.character(x), split = ",")[[1]]
  ifelse(length(x) > 1,
         as.numeric(paste(x, collapse=".")),
         as.numeric(x))
})
```

- **3a.** Convert the Wind column to numeric using `factor.to.numeric()`. Hint: use `mutate_at()`, and reassign `sprint.w.df` to be the output.

```
sprint.w.df <- sprint.w.df %>%
  mutate_at("Wind", factor.to.numeric)
```

```
## Warning in ifelse(length(x) > 1, as.numeric(paste(x, collapse = ".")),
## as.numeric(x)): NAs introduced by coercion
```

- **3b.** Run a linear regression of the Time on Wind columns, but only using data where Wind values that are nonpositive, and report the coefficients. Hint: use `filter()`, and use the dot `.` to pipe into the `lm()` function appropriately.

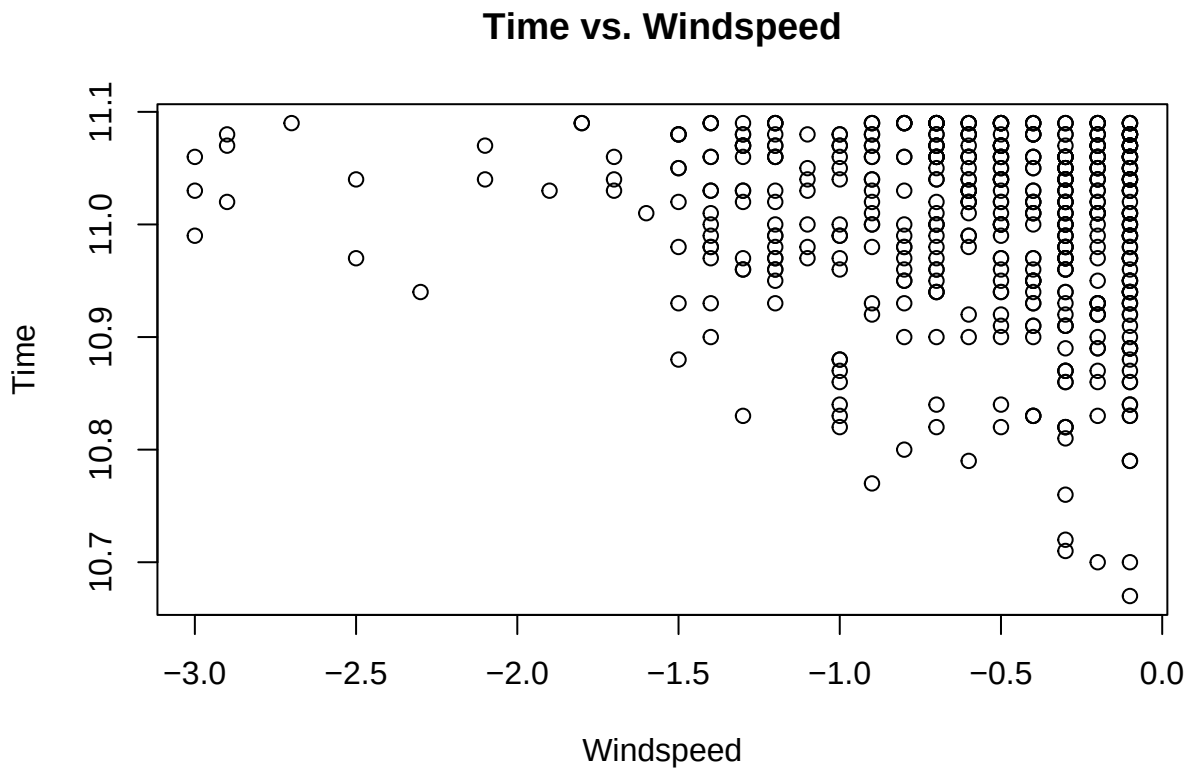
```
sprint.w.df %>%
  filter(Wind<0) %>%
```

```
  lm(Time ~ Wind, data = .) %>%
  coef
```

```
## (Intercept)        Wind
## 10.98967275 -0.02250043
```

- **3c.** Plot the Time versus Wind columns, but only using data where Wind values that are nonpositive, and label the axes appropriately. Hint: recall that for a data frame, with columns `colX` and `colY`, you can use `plot(colY ~ colX, data=df)`, to plot `df$colY` versus `df$colX`.

```
sprint.w.df %>%
  filter(Wind<0) %>%
  plot(Time ~ Wind, data=., main="Time vs. Windspeed", xlab="Windspeed", ylab="Time")
```



- **3d.** Reorder the rows in terms of increasing Wind, and then display only the women who ran at most 10.7 seconds. Hint: do this with one single flow of pipe commands; use `arrange()`, `filter()`.

```
sprint.w.df %>%
  arrange(Wind) %>%
  filter(Time<=10.7)
```

```
##    Rank  Time Wind                    Name Country Birthdate
## 1     2 10.70 -0.2     Svetlana Goncharenko     RUS  26.05.71
## 2     6 10.67 -0.1          Carmelita Jeter     USA  24.11.79
## 3     7 10.70 -0.1             Marion Jones     USA  12.10.75
## 4     1 10.49  0.0 Florence Griffith-Joyner     USA  21.12.59
## 5     1 10.60  0.0             Zhanna Block     UKR  06.07.72
## 6     2 10.70  0.0             Zhanna Block     UKR  06.07.72
```

```
## 7     7 10.70  0.3          Elaine Thompson     JAM  28.06.92
## 8     7 10.70  0.6  Shelly-Ann Fraser-Pryce     JAM  27.12.86
## 9     3 10.62  1.0 Florence Griffith-Joyner     USA  21.12.59
## 10    5 10.65  1.1             Marion Jones     USA  12.10.75
## 11    2 10.70  1.1          Juliet Cuthbert     JAM  09.04.64
## 12    2 10.61  1.2 Florence Griffith-Joyner     USA  21.12.59
## 13    4 10.64  1.2          Carmelita Jeter     USA  24.11.79
## 14    2 10.70  1.3        Blessing Okagbare     NGR  10.09.88
## 15    7 10.70  1.6 Florence Griffith-Joyner     USA  21.12.59
## 16    7 10.70  2.0          Carmelita Jeter     USA  24.11.79
##                   City       Date
## 1       Rostov-na-Donu 30.05.1998
## 2  Thessalon&iacute;ki 13.09.2009
## 3              Sevilla 22.08.1999
## 4         Indianapolis 16.07.1988
## 5                 Kiev 12.06.1997
## 6                 Kiev 12.06.1997
## 7             Kingston 01.07.2016
## 8             Kingston 29.06.2012
## 9                Seoul 24.09.1988
## 10        Johannesburg 12.09.1998
## 11            Kingston 04.07.1992
## 12        Indianapolis 17.07.1988
## 13            Shanghai 20.09.2009
## 14             El Paso 10.04.2010
## 15        Indianapolis 17.07.1988
## 16              Eugene 04.06.2011
```

- **3e.** Now reorder the rows in terms of increasing Time, and *then* increasing Wind, and again display only the women who ran at most 10.7 seconds, but only display the Time, Wind, Name, and Date columns. Hint: a single flow of pipe commands will do; note that `arrange()` can take multiple columns that you want to sort by, and the order you pass them specifies the priority.

```
sprint.w.df %>%
  arrange(Time, Wind) %>%
  filter(Time <= 10.7) %>%
  select(Time, Wind, Name, Date)
```

```
##     Time Wind                     Name       Date
## 1  10.49  0.0 Florence Griffith-Joyner 16.07.1988
## 2  10.60  0.0             Zhanna Block 12.06.1997
## 3  10.61  1.2 Florence Griffith-Joyner 17.07.1988
## 4  10.62  1.0 Florence Griffith-Joyner 24.09.1988
## 5  10.64  1.2          Carmelita Jeter 20.09.2009
## 6  10.65  1.1             Marion Jones 12.09.1998
## 7  10.67 -0.1          Carmelita Jeter 13.09.2009
## 8  10.70 -0.2     Svetlana Goncharenko 30.05.1998
## 9  10.70 -0.1             Marion Jones 22.08.1999
## 10 10.70  0.0             Zhanna Block 12.06.1997
## 11 10.70  0.3          Elaine Thompson 01.07.2016
## 12 10.70  0.6  Shelly-Ann Fraser-Pryce 29.06.2012
## 13 10.70  1.1          Juliet Cuthbert 04.07.1992
## 14 10.70  1.3        Blessing Okagbare 10.04.2010
## 15 10.70  1.6 Florence Griffith-Joyner 17.07.1988
## 16 10.70  2.0          Carmelita Jeter 04.06.2011
```

Prostate cancer data, revisited

Below we read in a data frame `pros.df` containing measurements on men with prostate cancer, as seen in previous labs. As before, in what follows, use `dplyr` and pipes to answer the following questions on `pros.df`.

```
pros.df =
  read.table("http://www.stat.cmu.edu/~ryantibs/statcomp/data/pros.dat")
```

- **4a.** Among the men whose lcp value is equal to the minimum value, report the lowest and highest lpsa score.

```
pros.df %>%
  filter(lcp==min(lcp)) %>%
  filter(lpsa==min(lpsa) | lpsa==max(lpsa))
```

```
##        lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1  -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 92  2.5329028 3.677566  61  1.348073   1 -1.386294       7    15  4.1295508
```

- **4b.** Order the rows by decreasing age, then decreasing lpsa score, and display the rows from men who are older than 70, but only the age, lpsa, lcavol, and lweight columns.

```
pros.df %>%
  arrange(desc(age), desc(lpsa)) %>%
  filter(age>70) %>%
  select(age, lpsa, lcavol, lweight)
```

```
##    age       lpsa     lcavol  lweight
## 47  79  2.5687881  2.7278528 3.995445
## 78  78  3.4355988  2.5376572 4.354784
## 83  77  3.5652984  2.6130067 3.888754
## 72  77  3.0373539  1.1600209 3.341093
## 90  76  3.9936030  1.5623463 3.695110
## 3   74 -0.1625189 -0.5108256 2.691243
## 61  73  2.8419982  0.4574248 4.524502
## 37  73  2.1575593  1.4231083 3.657131
## 77  72  3.3928291  2.0108950 4.433789
## 70  72  2.9729753  1.1939225 4.780383
## 68  72  2.9626924  2.1983351 4.050915
## 63  72  2.8535925  2.7757089 3.524889
## 33  71  2.0082140  1.2753628 3.037354
```

- **4c.** Run a linear regression of the lpsa on lcavol and lweight columns, but only using men whose lcp value is strictly larger than the minimum value, and report a summary of the fitted model.

```
pros.df %>%
  filter(lcp>min(lcp)) %>%
  lm(lpsa ~ lcavol + lweight, data = .) %>%
  summary
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight, data = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.69447 -0.39524  0.06257  0.33075  1.87082
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6699     1.0512   0.637    0.527
## lcavol        0.6462     0.1221   5.291 2.83e-06 ***
## lweight       0.2846     0.2907   0.979    0.332
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.787 on 49 degrees of freedom
## Multiple R-squared:  0.4013, Adjusted R-squared:  0.3769
## F-statistic: 16.42 on 2 and 49 DF,  p-value: 3.48e-06
```

- **4d.** Extend your code in the last part, still just using a single flow of pipe commands in total, to extract the p-values associated with each of the coefficients in the fitted model.

```
pros.df %>%
  filter(lcp>min(lcp)) %>%
  lm(lpsa ~ lcavol + lweight, data = .) %>%
  summary %>%
  .$coefficients %>%
  .[,"Pr(>|t|)"]
```

```
##  (Intercept)       lcavol      lweight
## 5.268747e-01 2.832674e-06 3.322792e-01
```