Coding Help Sheet 3

tidyverse and ggplot2

Fred Hutch Cancer Center

Filtering Dataframes

filter() lets you select a subset of rows in a data frame

```
# Filter by columns 1 and 2
data %>%
  filter(column2==”values”,
       column1 > 3)
```

arrange() sorts observations by ascending or descending order

```
# Sorts data by ascending order of column2 length
data %>%
  arrange(column2.length)

# Sorts by descending order
data %>%
   arrange(desc(column2.length))
```

You can also **combine dplyr verbs** in a row with the pipe operator

```
data %>%
  filter(column2==”values”,column1 > 3) %>%
  arrange(desc(column2.length))
```

Summarizing Dataframes

summarize() allows you to turn observations into a single data point

```
# Summarize to find median val length
data %>%
  summarize(medianVal = median(value.length))
```

tidyverse Functionality

tidyverse uses a **split-apply-combine** methodology to break up problems into separate tasks. We use functions to perform these tasks one by one:

```
# For example, taking the average of y for each category in x
my_tibble %>%
   group_by(x) %>%
   mutate(y = avg(y)) %>%
   ungroup()
```

dplyr Functions

group_by(v)	Groups dataframe by v or summarize within groups
tally()	Returns tally of given value
summarize(c1, c2)	summarize the entire dataset given multiple columns (c1, c2...)
max(x.length) min(x.length)	Returns the max and min length of column x
count()	Shorthand for group_by() and tally()
%in%	pass vector to filter against (same as equals OR)
select(df, c1, c2)	selects specific columns from dataframe df (c1, c2)
mutate(newCol = c1 + c2)	create or modifies existing dataframe by adding a new column based on previous columns

ggplot2 library

Below are common ways to extract relevant data from data frames using [ ] notation.

ggsave(path = “filepath”, width=1, height=1)	Save plot in a file (can be .png or other valid file formats with given width and height of image)
ggplot(data = clinical)	binds given dataframe (clinical) to the plot
ggplot(data = clinical, mapping =aes(x=xaxis_name, y=yaxis_name))	specifies axes names in ggplot using mapping() and aes()
ggplot(..) + geom_point()	Add layer of geometry to plot
ggplot() + coord_cartesian(xlim = c(0, 10), ylim = c(0, 400))	specify the x (using the argument **xlim**) and y (using the argument **ylim**) ranges of interest
geom_jitter(color = "blue")	Add jitter points with specific color indicated

What makes a good plot?

Choose the plot most **appropriate** for the **data** and the **goal**

Clear axis **labels**

Include a **legend** (if informative)

Plot colors are visually **appealing**, **informative**, and **accommodating** to all

Text in the plot are **clear and legible**

Types of Plots

Barplots are effective at making **numerical comparisons** (usually counts) between **distinct categories.**

Histograms are useful to **understand the range (or ‘spread’)** of a set of **numerical** values.

Boxplots, like barplots, allow us to **numerically compare** data between different **categories**. In addition, it also gives us information about the **range of the data** in each category (like histograms).

Scatterplots are useful to examine the relationship between **two continuous numerical variables**. For more info.