Basic R Functions

By Brianna C. Heggeseth + Students

Reading in Data (readr package)

`dat = read_csv()`: reads in a csv file into R and names it Dat
`dat = read_delim()`: reads in a delimited file into R and names it Dat
`dat = read_table()`: reads in a text file into R and names it Dat
`require(foreign)`: this package includes functions to read in data from other stat software (STATA, etc)
`load()`: loads data and objects from an .RData file

Numerical Summary Functions (dplyr package)

`dat %>% count(xvar)`: creates a frequency table for categorical xvar
`dat %>% count(xvar,yvar)`: creates a contingency table for categorical xvar and yvar
`dat %>% count(xvar) %>% mutate(relfreq = n/sum(n))`: calculates relative frequencies given a frequency table tab
`dat %>% count(xvar,yvar) %>% group_by(xvar) %>% mutate(relfreq = n/sum(n))`: calculates conditional relative frequencies, conditional on xvar
`dat %>% count(xvar,yvar) %>% group_by(yvar) %>% mutate(relfreq = n/sum(n))`: calculates conditional relative frequencies, conditional on yvar

`dat %>% select(xvar) %>% summary()`: calculates min, Q1, median, mean, Q3, and na.rm max for quantitative variable xvar
`dat %>% summarize(m = mean(xvar, na.rm=T))`: calculates mean for quantitative variable xvar
`dat %>% summarize(m = median(xvar, na.rm=T))`: calculates median for quantitative variable xvar
`dat %>% summarize(iqr =IQR(xvar, na.rm=T))`: calculates IQR for quantitative variable xvar
`dat %>% summarize(s = sd(xvar, na.rm=T))`: calculates SD for quantitative variable xvar
`c`: calculates correlation coefficient between two quantitative variables xvar and yvar

`dat %>% summarize_all(function(x) sum(is.na(x)))`: count the number of missing values for each variable in the dataset

Numerical Summary within Groups (dplyr package)

`dat %>% group_by(xvar) %>% summarise(m = mean(yvar, na.rm=T))`: returns the means of yvar within each group defined by xvar
`dat %>% group_by(xvar) %>% summarise(m = median(yvar, na.rm=T))`: returns the medians of yvar within each group defined by xvar

`dat %>% group_by(xvar) %>% summarise(s = sd(yvar, na.rm=T))`: returns the standard deviations of yvar within each group defined by xvar

ggplot2 Plotting Functions

`ggplot(dat) + geom_bar(aes(x = xvar))`: creates a barplot given a categorical variable xvar in dat

`ggplot(dat) + geom_histogram(aes(x = xvar), bins)`: creates a histogram given a quantitative variable xvar in dat

`ggplot(dat) + geom_boxplot(aes(x=1,y=yvar))`: creates a boxplot given a quantitative variable yvar in dat

`ggplot(dat) + geom_boxplot(aes(x=xvar,y=yvar))`: creates multiple boxplots of quantitative variable yvar across groups xvar in dat

`ggplot(dat) + geom_point(aes(x=xvar,y=yvar))`: creates a scatterplot given quantitative variables xvar and yvar in dat

`ggplot(dat, aes(x=xvar,y=yvar)) + geom_point(color='blue') + geom_smooth(method='lm',se=FALSE)`: creates a scatterplot given quantitative variables xvar and yvar in dat and adds a least squares line (based on lm())

`dat %>% select(var1, var2, var3) %>% GGally::ggpairs()` : Gives pairwise comparison of each variable.

ggplot2 Plotting Functions -- Making things pretty

`+ theme_classic()`: removes gray background and simplifies axes

`+ theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))`: rotates x-axis labels 90 degrees

`+ labs(x = 'x label', y = 'y label', fill = 'fill label', title = 'Title')`: changes x-axis label, y-axis label, the legend label, and the title

`+ scale_fill_viridis_d()`: if you are using a fill input in aes(), you can change the colors to be color-blind friendly

`+ scale_x_continuous(name = "New Label", labels = scales::comma)`: change axis tick mark labels to say, for example, "400,000" instead of "4e+05"

For more info about scales and colors, see:
https://rdpeng.github.io/RProgDA/customizing-ggplot2-plots.html#scales-and-color

Subsetting Functions (Tidyverse)

`dat %>% slice(1:10)`: returns the first 10 rows

`dat %>% filter(xvar > 10)`: return in s the rows for which xvar is greater than 10

`dat %>% filter(xvar == 10)`: returns the rows for which xvar is equal to 10

`dat %>% filter(xvar %in% c('cat1', 'cat2', 'cat3'))`: returns the rows for which the value for xvar is in a list of possible options (i.e., xvar is equal to either "cat1" or "cat2" or "cat3")
`dat %>% filter(!is.na(var1), !is.na(var2))`: returns the rows which are *not* missing var1 and var2
`dat %>% select(xvar)`: returns the column with name xvar
`dat %>% select(xvar1,xvar2,xvar3)]`: returns the columns with names xvar1, xvar2, xvar3

`dat %>% select(xvar1, xvar2) %>% na.omit() %>% ggplot()`: to temporarily remove missing values from the two selected variables
F
`dat %>% count(catvar)`: to check the number of missing values

Merging Datasets (Tidyverse)
`newdat = left_join(dat1,dat2)`: merge two data set by the variables with exactly the same names in both datasets, keeping all of the rows in dat1, no matter if they have a match in dat2
`newdat = left_join(dat1,dat2,by=c('id'= 'ID'))`: merge two data set by the variables identified in by, keeping all of the rows in dat1, no matter if they have a match in dat2

Reformatting Datasets from Wide to Long and Long to Wide (Tidyverse)

https://tidyr.tidyverse.org/

```
gather(dat, key = year, value = fertility, -VarstoBeExcluded)
```

Recoding Categorical Variables (Tidyverse)

`dat <- dat %>% mutate(var = relevel(droplevels(var), ref = "newreference"))`: Change the reference level of a categorical variable and remove any categories that aren't present in the data.dr

Note: to see the levels of the categorical variable, use `levels(dat$var)`

`dat <- dat %>% mutate(var = factor(var, levels=c('cat1','cat2','cat3','cat4'))`: Change the order of the categories of a categorical variable (cat1 through cat4).

`dat <- dat %>% mutate(var.new = recode(var, `oldcat1` = "newcat1", `oldcat2` = "newcat2", `oldcat3` = "newcat3" ))`: Change the labels of a categorical variable

```
dat <- dat %>% mutate(var.new = case_when(
     Tvar == 'oldcat1' ~ "newcat1",
     var == 'oldcat2' ~ "newcat2",
     var == 'oldcat3' = "newcat3" ))
```
: Change the labels of a categorical variable

```
dat <- dat %>%
     mutate(var.new = if_else(var == 'oldcat1', "newcat1", var)) %>%
     mutate(var.new = if_else(var == 'oldcat2', "newcat2", var)) %>%
     mutate(var.new = if_else(var == 'oldcat3', "newcat3", var))
```
: Change the labels of a categorical variable

```
dat <- dat %>% mutate(var.new = case_when(
     var %in% c('oldcat1', 'oldcat2') ~ "newcat1",
     var %in% c('oldcat3', 'oldcat4', 'oldcat5') ~ "newcat2",
     var %in% c('oldcat6', 'oldcat7') ~ "newcat3" ))
```
: combine categories together into fewer categories

Recoding Quantitative Variable as Categorical Variable (Tidyverse)

`dat <- dat %>% mutate(varcat = cut(var, breaks=c(-Inf, 0.5, 0.6, Inf), labels=c("low","middle","high")))`: Convert a quantitative variable into a categorical variable. Breaks give the endpoints of the intervals. Labels give the name of the category.

`dat <- dat %>% mutate(binaryvar = quantvar >= 10)`: Convert a quantitative variable into a binary variable by checking to see if it is bigger than 10 or not. The new binary variable will be either TRUE (1) or FALSE (0).

https://forcats.tidyverse.org/reference/fct_reorder.html

Linear Regression Models (broom package)

`1 = dat %>% with(lm(y~x1+x2))`: gives you estimated slopes (coefficients) that minimize the sum of squared residuals based on a linear model.

`tidy(lm1)`: gives you a tidy version of information about the line (estimates, standard errors, test statistics, p-values)

Linear Model Evaluation (broom package)

`glance(lm1)`: gives you R2 and standard deviation of residuals (sigma)

`augment(lm1)`: gives you a data frame with the variables used in the model, the predicted (.fitted) values, yhat, the residuals (.resid), etc. This is useful to create residual plots.

`predict(lm1,newdata=data.frame(x1=1,x2=4))`: gives you the predicted value of y for a new value of x values.

Logistic Regression Models (broom package)

`glm1 = dat %>% with(glm(y~x1+x2, family=binomial))`: gives you estimated slopes (coefficients) from a logistic regression model with a binary (0/1 or FALSE/TRUE) y outcome variable.

`tidy(glm1)`: gives you a tidy version of information about the line (estimates, standard errors, test statistics, p-values)

Logistic Model Evaluation (broom package)

`glance(glm1)`: not useful for this class

`augment(glm1, type.predict ='response')`: gives you a data frame with the variables used in the model, the predicted probabilities (`.fitted`), etc. This is useful to create predicted probability boxplots

`predict(glm1,newdata=data.frame(x1=1,x2=2), type.predict ='response')`: gives you the predicted value of y for a new value of x values.

Inference

`tidy(mod)`: gives you summary output of test statistics and p-values for regression slopes

`confint(mod)`: gives you a 95% CI for the regression slopes

`anova(mod1,mod2)`: performs a nested F test between two nested regression models

`anova(mod1,mod2, test = 'LRT')`: performs a nested test between two nested logistic regression models

Probability Models

`rnorm(),rbinom(),rt(),rchisq()`: randomly generates data from probability model

`pnorm(),pt(),pchisq()`: gives the cdf (area to the left) - used to find p-values

`qnorm(),qt()`: gives the inverse cdf (point on the line for a given area to the left) - used to critical values for confidence intervals