STA 610/615 Week 01

D. Zeitler

August, 2017

Get some packages installed/loaded. *library(tidyverse)* *bold*

library(tidyverse)

## -- Attaching packages -------------------------------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.0.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.6  
## v tidyr 0.8.1 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## -- Conflicts ----------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

#install.packages('ggformula')  
library(mosaic)

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggstance

##   
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':  
##   
## geom\_errorbarh, GeomErrorbarh

##   
## New to ggformula? Try the tutorials:   
## learnr::run\_tutorial("introduction", package = "ggformula")  
## learnr::run\_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following object is masked from 'package:tidyr':  
##   
## expand

##   
## The 'mosaic' package masks several functions from core packages in order to add   
## additional features. The original behavior of these functions should not be affected by this.  
##   
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##   
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':  
##   
## mean

## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally

## The following object is masked from 'package:purrr':  
##   
## cross

## The following object is masked from 'package:ggplot2':  
##   
## stat

## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cor.test, cov, fivenum, IQR, median,  
## prop.test, quantile, sd, t.test, var

## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

library(ggformula)

We might want to change some preferences in RStudio.

Preferences

Preferences

Now let’s take a look at a data file. We’ll start with Fisher’s\_Iris.xls from Moodle.

#install.packages('readxl')  
library(readxl)  
Fisher\_s\_Iris <- read\_excel("Fishers\_Iris.xls",  
 skip = 3,  
 col\_names = c('SpeciesNum','Species',  
 'PetalWidth','PetalLength',  
 'SepalLength', 'SepalWidth')  
 )  
Fisher\_s\_Iris

## # A tibble: 150 x 6  
## SpeciesNum Species PetalWidth PetalLength SepalLength SepalWidth  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 1 I. Setosa 2 14 33 50  
## 2 3 I. Verginica 24 56 31 67  
## 3 3 I. Verginica 23 51 31 69  
## 4 1 I. Setosa 2 10 36 46  
## 5 3 I. Verginica 20 52 30 65  
## 6 3 I. Verginica 19 51 27 58  
## 7 2 I. Versicolor 13 45 28 57  
## 8 2 I. Versicolor 16 47 33 63  
## 9 3 I. Verginica 17 45 25 49  
## 10 2 I. Versicolor 14 47 32 70  
## # ... with 140 more rows

We can save data frames in CSV format.

They are then easily read back in.

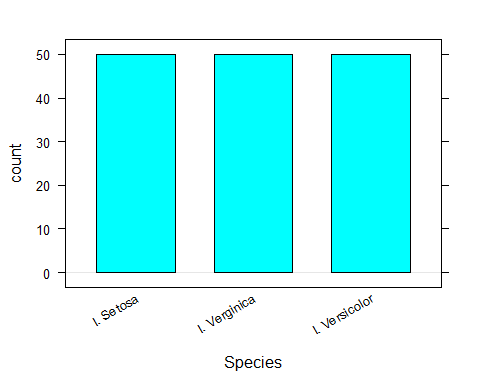
#fi <- read\_csv('FishersIris.csv')  
#read\_csv('FishersIris.csv') -> fi  
#read\_csv('FishersIris.csv') %>% summary()

The mosaic package.

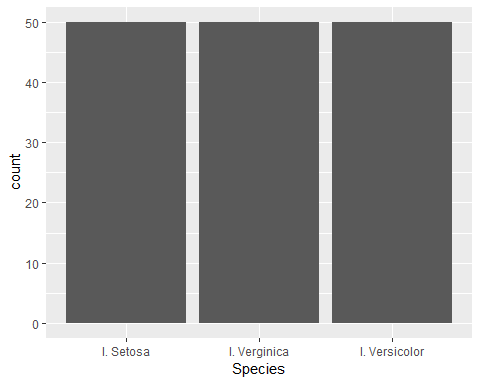
goal(y~x|z, data=…, groups=…) y - dependent variable or vertical axis x - independent variable or horizontal axis z - conditioning variable (panels) groups - conditioning variable (overlay)

Let’s try it.

mosaic::bargraph(~Species,data=Fisher\_s\_Iris)

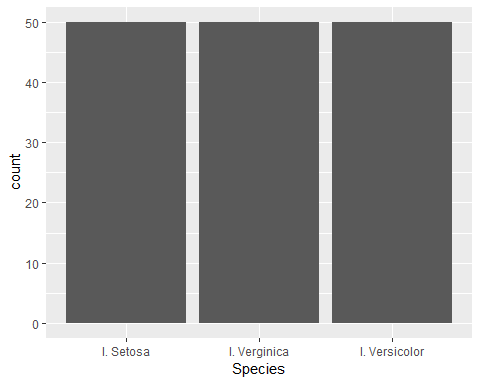


gf\_bar(~Species,data=Fisher\_s\_Iris)



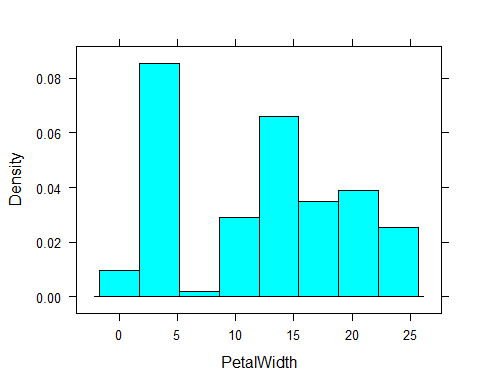
Alternatively, using the magittr pipes.

Fisher\_s\_Iris %>% gf\_bar(~Species)

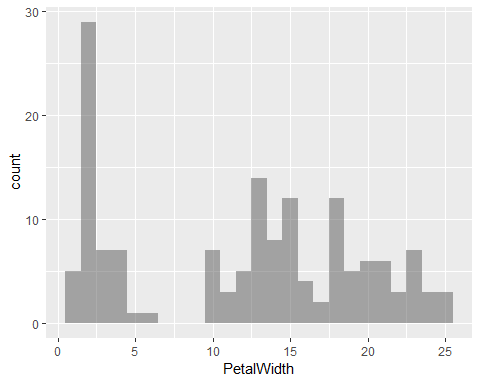


Something more interesting. Note we can drop the ‘data =’ if the order of paramters in the function call match their defined order. It’s probably a good idea to get in the habit of explicitely using ‘data =’.

histogram(~PetalWidth,data=Fisher\_s\_Iris)



Fisher\_s\_Iris %>% gf\_histogram(~PetalWidth)



mean(~PetalWidth,data=Fisher\_s\_Iris)

## [1] 11.92667

sd(~PetalWidth,data=Fisher\_s\_Iris)

## [1] 7.569008

favstats(~PetalWidth,data=Fisher\_s\_Iris)

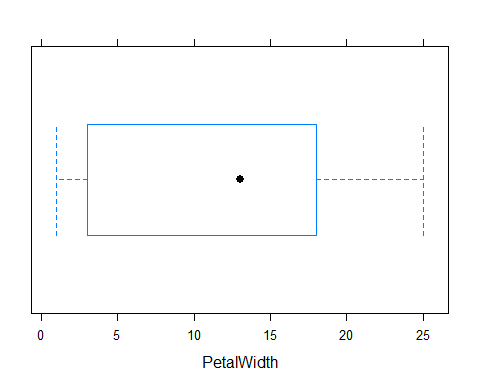
## min Q1 median Q3 max mean sd n missing  
## 1 3 13 18 25 11.92667 7.569008 150 0

favstats(PetalWidth~Species,data=Fisher\_s\_Iris)

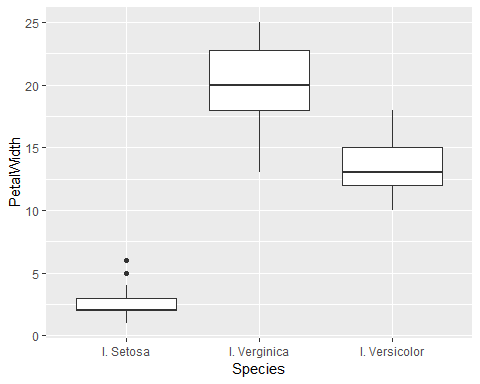
## Species min Q1 median Q3 max mean sd n missing  
## 1 I. Setosa 1 2 2 3.00 6 2.46 1.053856 50 0  
## 2 I. Verginica 13 18 20 22.75 25 20.06 2.902567 50 0  
## 3 I. Versicolor 10 12 13 15.00 18 13.26 1.977527 50 0

Now let’s get a box plot of exactly the same data.

bwplot(~PetalWidth,data=Fisher\_s\_Iris)

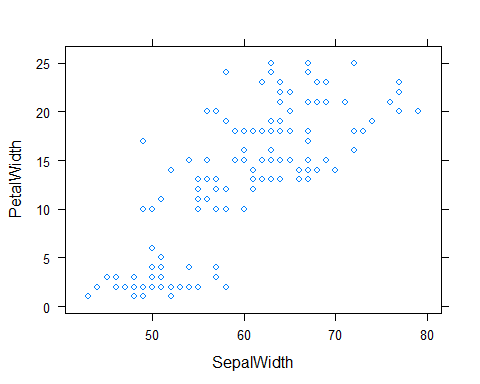


# Note single box plots with ggformula currently doesn't work as expected  
#Fisher\_s\_Iris %>% gf\_boxplot(~PetalWidth)  
# We can get it with this, but I prefer bwplot.  
Fisher\_s\_Iris %>% gf\_boxplot(PetalWidth~Species)

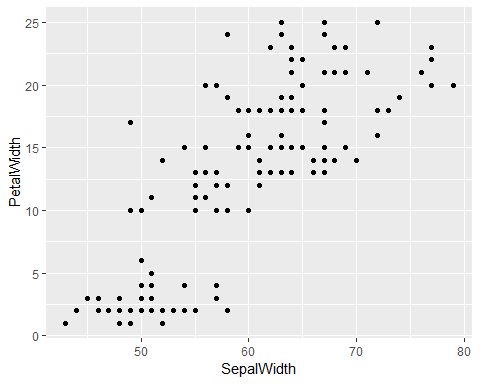


# Actually, single sample box plots are rather useless. Use histograms instead.

xyplot( PetalWidth ~ SepalWidth, data = Fisher\_s\_Iris)

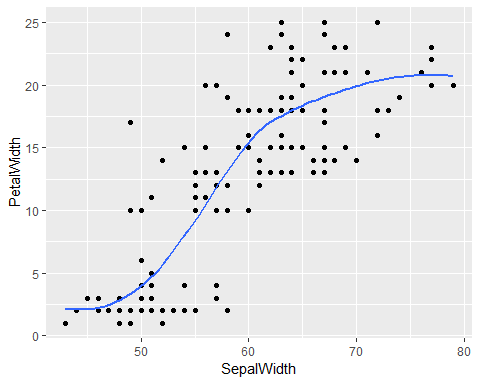


gf\_point(Fisher\_s\_Iris, PetalWidth ~ SepalWidth)



gf\_point(Fisher\_s\_Iris, PetalWidth ~ SepalWidth) %>%  
 gf\_smooth()

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



gf\_point(Fisher\_s\_Iris, PetalWidth ~ SepalWidth) %>%  
 gf\_smooth(se=T)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

