Markdown Example

Your Name

January ??, 2024

Contents

Last updated on: January 09, 2024 at 19:33

Instructions

After installing R and RStudio using these instructions start RStudio in the project you created for this course.

Use the menus to create a new R script with *File > New File > R Script.*

Copy this file into the R Script and save it with *Control+S* (*Command+S* in macOS) with the name 'markdown_sample.R'

Execute this file manually, line by line, using *Control+Enter* in Windows or *Command+Enter* on macOS. As you go through the file, make appropriate changes: e.g. fill in your name on line 3 above.

This will install a number of packages and download a data file.

After executing the file line by line you should be able to 'render' it to create a stand-alone HTML file by pressing *Control+Shift+K* (*Command+Shift+K* on macOS).

R Markdown and Reproducible Research

This is an example of output from a ".R script with R Markdown". If you run this script in R, all lines that begin with '#' are treated as comments. If you run it with 'knitr' – which just involves typing *Control-Shift-K* simultaneously when the file is the active file in R Studio – all the text in lines that begin with "#' " (hashtag apostrophe space) are processed as R Markdown code that can be used to produce a polished document in a number of standard formats: pdf, HTML, RTF, beamer presentations, etc.

Note that online help for R Markdown describes syntax for ".Rmd" files. They are the same as ".R scripts with R Markdown" files minus the "#' " and plus *code chunk delimiters*. In .R scripts, the code chunk delimiters are not necessary, which makes .R scripts easier to use. There are legitimate reasons to prefer ".Rmd" files but for uniformity when collaborating it is suggested that you use ".R scripts with R Markdown" in this course.

Almost all the work in this course will be done and submitted this way.

Advantages of R scripts: Reproducible Analyses

One of the great strengths of Markdown is that it allows you to do **reproducible** analyses and reports. Another researcher can use your R script on the same data and get the same results. They can verify the exact steps you took and can easily test how results would be affected by modifying the analysis.

You can collaborate with others much more easily knowing that if you send a script and data files to a collaborator they will get the same output (provided they have installed the same packages).

An important practice for reproducible research is that the analyst should never modify raw data. For example, if you find some errors in a spreadsheet sent by a client or downloaded from the internet, it is very tempting to manually correct the errors in the spreadsheet.

However, when you receive or download a new version of the spreadsheet you would need to apply the corrections the same way. If you did them manually you will not be able to guarantee consistency in your corrections.

To ensure reproducibility, the corrections should be done with code in the R script. This will usually involve the use of a very powerful tool: regular expressions, the subject of this xkcd cartoon.

With regular expressions, data corrections can usually be made in a way that does not introduce errors when applied to a future corrected version of the spreadsheet.

Advantages of R Markdown: Simple markup and LaTeX formulas

As illustrated in this script, you can combine:

1. R code
2. R output
3. R graphics
4. text
5. hyperlinks
6. embedded graphics
7. headings, table of contents and other markup
8. mathematical formulas in LaTeX
9. and, with a bit of effort, interactive graphics and other widgets

in a document.

LaTeX is the most widely used language and environment for technical publishing.
In R Markdown, we have access to its language for mathematical formulas. For example, we can write:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \quad i = 1, ..., n, \; \epsilon \sim \mathrm{N}(0, \sigma^2)$$

where $\beta_0 = 2$, $\beta_1 = 0.5$, $\sigma = 2$.

All you need in order to use R Markdown to produce HTML output is automatically installed when you install R Studio.

Installing R and R Studio

If you are running this script, you have probably already installed R and RStudio.

To install R visit this ‘mirror’ of the The Comprehensive R Archive Network and follow the instructions for your operating system: Mac OS X, Windows or Linux.

Read the information carefully. If you use Windows, install ‘Rtools’ as suggested. If you use Mac OS X, consider whether you need to install XQuartz.

After installing R, next install R Studio.

Once R Studio is installed, open it with the icon on your desktop. We will use the console in R Studio to install some packages.

Installing and loading packages

There are three main sources of packages for R:

1. Many packages come with R when you install it initially.
2. Most additional packages you might consider installing reside on ‘CRAN’, the Comprehensive R Archive Network, that provides relative stable versions of packages that have passed some automated tests, and
3. ‘github.com’ where most developers provide access to the latest beta versions of their packages or to packages that have not been sent for inclusion in CRAN.

The first package we install resides on CRAN and it is needed to install packages from GitHub.

Note that “#+ ” denotes a “chunk option” that modifies the behaviour of R until the end of the chunk. A chunk is ended by any line that starts with “#' ” or with “#+ ”. The option “”’eval=FALSE” prevents R from running (evaluating) the chunk when you use Crtl-Shift-K to produce an output file. We assume that all packages have already been installed interactively before using Crtl-Shift-K.

If you haven’t installed the following packages yet, this is an opportunity to do so. You must run these lines ‘manually’ by using Ctrl-Enter with the cursor on the line.

```r
# These lines will not be run when you render the script with Ctrl-Shift-K
install.packages('devtools')
```

While we are at it we can install a few more packages from CRAN:

```r
install.packages(c("car", "effects", "ggplot2", "Hmisc"))
install.packages(c("knitr", "magrittr", "rgl", "rio", "rmarkdown", "readxl", "cv"))
install.packages(c("latticeExtra"))
install.packages('kableExtra')
devtools::install_github('gmonette/spida2')
devtools::install_github('gmonette/p3d')
```

Installing packages only needs to be done once every time you install a new version of R. You can update them occasionally, with:

```r
update.packages()
```

Github packages need to be reinstalled periodically to get the latest updates:

```r
# devtools::install_github('gmonette/spida2')
# devtools::install_github('gmonette/p3d')
```

I expect these packages to be updated frequently during the course and you need to reinstall them to have access to the latest versions.

Each time you use R, you need to load the packages you need for that session with the ‘library’ command:

```
library(car)
```

```
Loading required package: carData
```

```
library(spida2)
library(lattice)
library(latticeExtra)
```

Reading data in a text file

The easiest formats to read are CSV files (comma-separated-value files which are easily created from Excel by saving a spreadsheet as a CSV file), and tab-delimited text files, usually with a '.txt' extension.

You need to know the location of the file relative to the script file.

Be sure to set the working directory, which you can find with

```
getwd()
```

```
[1] "/home/georges/4939/www/files"
```

to the directory of the script file. In RStudio, use the menus:

Session > Set Working Directory > To Source File Location

Then you can refer to the file using a relative path. This is much better than an absolute path since you can send the script and the data set to a collaborator and they will be able to run the script after saving the script and the data set in a directory on their own computer.

To illustrate, we will download a data file and use for a brief exploration,

```
# You only need to download once, so this is in a code chunk that
# will not be run when you use Ctrl-Shift-K to render the file.
download.file("http://socserv.socsci.mcmaster.ca/jfox/Books/Applied-Regression-3E/datasets/Titanic.txt"
                 "Titanic.txt")
```

This copies the file 'Titanic.txt' into the working directory. This file is a list of passengers on the Titanic and does not include the crew.

Note that all the data sets from the textbook can be downloaded this way. However, it is better to create a 'data' subdirectory and download them there.

```
list.files()
```

```
 [1] "description.html"           "description.pdf"
 [3] "description.R"              "Intro"
 [5] "markdown_sample.pdf"        "markdown_sample.R"
 [7] "markdown_sample.spin.R"     "markdown_sample.spin.Rmd"
 [9] "software_installation.html" "software_installation.R"
[11] "Titanic.txt"
```

You can also read a text file from the internet by using the URL but then you need to be connected to the internet whenever you read the file.

```
titanic <- read.table("Titanic.txt", header = T)
library(spida2)
library(lattice)
head(titanic)  # first 6 lines
```

```
                                                survived     age passengerClass
Allen, Miss Elisabeth Walton                         yes 29.0000            1st
```

```
Allison, Miss Helen Loraine                           no  2.0000            1st
Allison, Mr Hudson Joshua Creighton                   no 30.0000            1st
Allison, Mrs Hudson J.C. (Bessie Waldo Daniels)       no 25.0000            1st
Allison, Master Hudson Trevor                        yes  0.9167            1st
Anderson, Mr Harry                                   yes 47.0000            1st
                                                   sex
Allen, Miss Elisabeth Walton                    female
Allison, Miss Helen Loraine                     female
Allison, Mr Hudson Joshua Creighton               male
Allison, Mrs Hudson J.C. (Bessie Waldo Daniels) female
Allison, Master Hudson Trevor                     male
Anderson, Mr Harry                                male
```

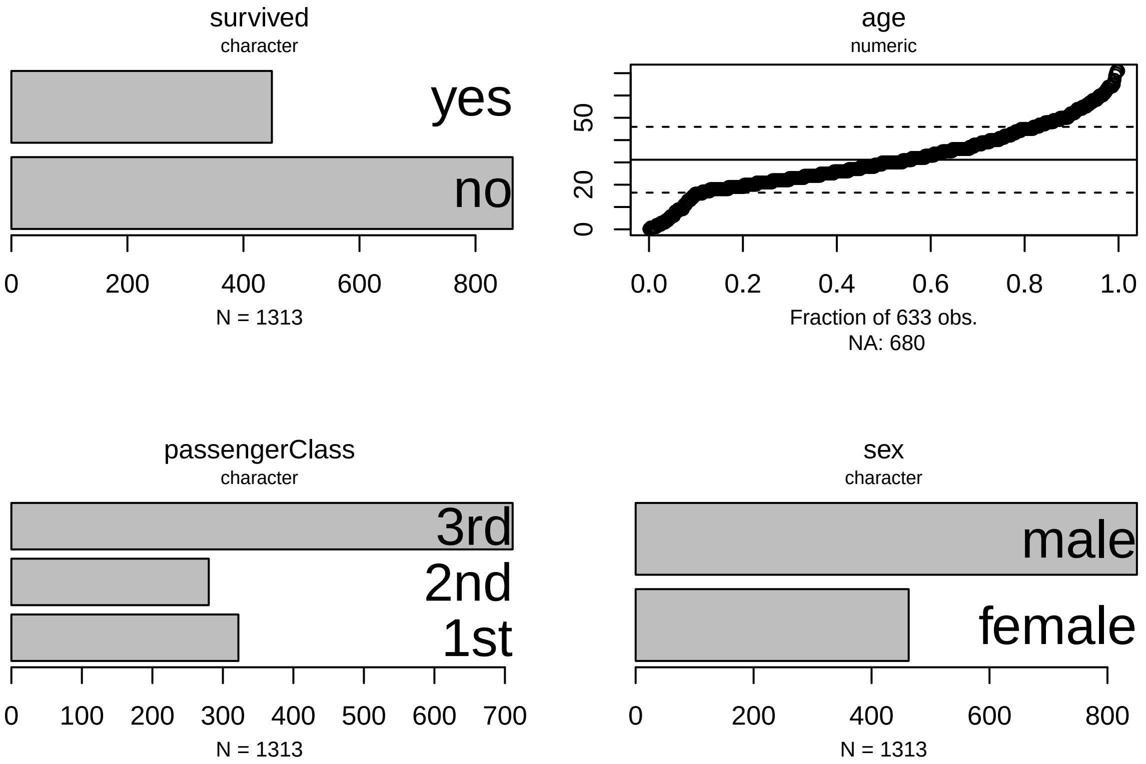
```r
tail(titanic)  # last 6 lines
```

```
                       survived age passengerClass    sex
Zabour, Miss Tamini          no  NA            3rd female
Zakarian, Mr Artun           no  NA            3rd   male
Zakarian, Mr Maprieder       no  NA            3rd   male
Zenn, Mr Philip              no  NA            3rd   male
Zievens, Rene                no  NA            3rd female
Zimmerman, Leo               no  NA            3rd   male
```

```r
dim(titanic)   # rows and columns
```

```
[1] 1313    4
```

```r
xqplot(titanic)
```



```r
#
# frequency table
#
```

```r
tab(titanic, ~ survived + sex + passengerClass)
```

```
, , passengerClass = 1st

        sex
survived female male Total
   no         9  120   129
   yes      134   59   193
   Total    143  179   322

, , passengerClass = 2nd

        sex
survived female male Total
   no        13  148   161
   yes       94   25   119
   Total    107  173   280

, , passengerClass = 3rd

        sex
survived female male Total
   no       134  440   574
   yes       79   58   137
   Total    213  498   711

, , passengerClass = Total

        sex
survived female male Total
   no       156  708   864
   yes      307  142   449
   Total    463  850  1313
```

```r
#
# percentage within each 'sex by passengerClass' grouping
#
tab(titanic, ~ survived + sex + passengerClass, pct = c(2,3))
```

```
, , passengerClass = 1st

        sex
survived     female       male        All
   no      6.293706  67.039106  40.062112
   yes    93.706294  32.960894  59.937888
   Total 100.000000 100.000000 100.000000

, , passengerClass = 2nd

        sex
survived     female       male        All
   no     12.149533  85.549133  57.500000
   yes    87.850467  14.450867  42.500000
   Total 100.000000 100.000000 100.000000
```

```
, , passengerClass = 3rd

        sex
survived     female       male        All
   no     62.910798  88.353414  80.731364
   yes    37.089202  11.646586  19.268636
   Total 100.000000 100.000000 100.000000

, , passengerClass = All

        sex
survived     female       male        All
   no     33.693305  83.294118  65.803503
   yes    66.306695  16.705882  34.196497
   Total 100.000000 100.000000 100.000000
```

```r
# nicer:
tab(titanic, ~ survived + sex + passengerClass, pct = c(2,3))  %>%
  round(1)
```

```
, , passengerClass = 1st

        sex
survived female  male   All
   no       6.3  67.0  40.1
   yes     93.7  33.0  59.9
   Total  100.0 100.0 100.0

, , passengerClass = 2nd

        sex
survived female  male   All
   no      12.1  85.5  57.5
   yes     87.9  14.5  42.5
   Total  100.0 100.0 100.0

, , passengerClass = 3rd

        sex
survived female  male   All
   no      62.9  88.4  80.7
   yes     37.1  11.6  19.3
   Total  100.0 100.0 100.0

, , passengerClass = All

        sex
survived female  male   All
   no      33.7  83.3  65.8
   yes     66.3  16.7  34.2
   Total  100.0 100.0 100.0
```

Note that the '%>%' operator 'pipes' the output of the left-hand side (lhs) as the first argument of the function on the right.

In RStudio, you can type '%>%' by pressing *Control-Shift-M*.

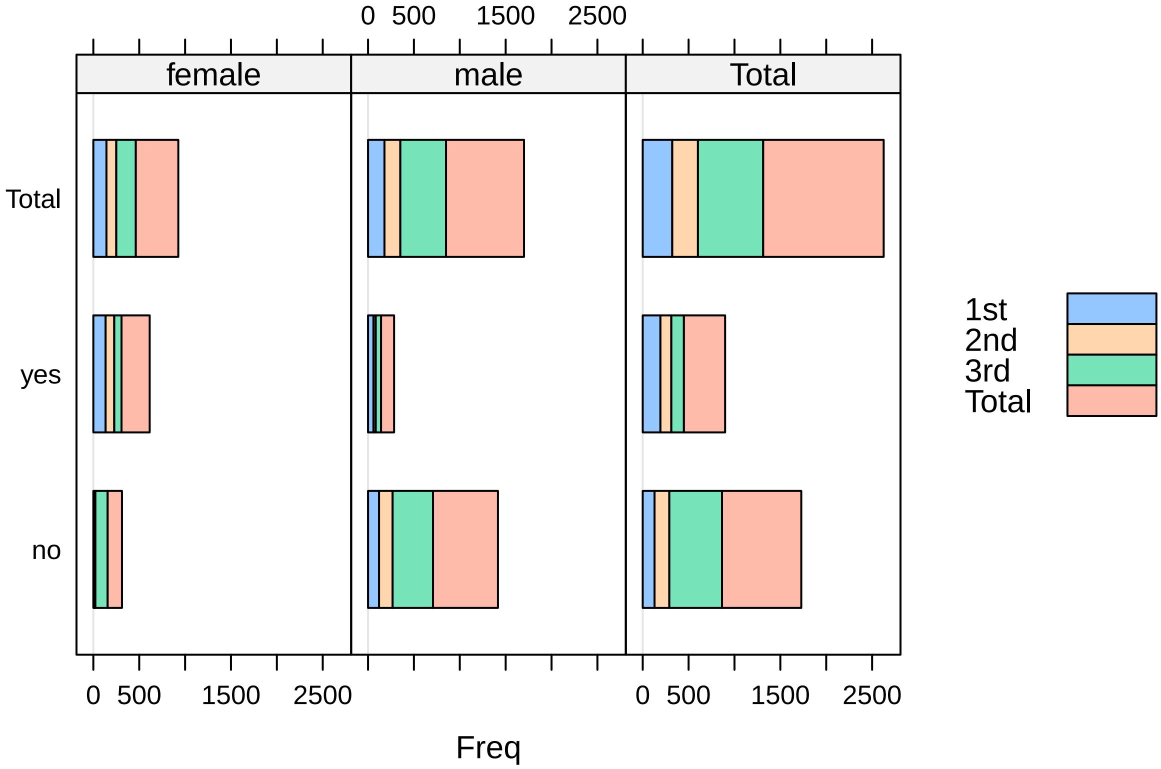
Visualizing frequencies

Barchart of frequencies

```
tab(titanic, ~ survived + sex + passengerClass)
```

```
, , passengerClass = 1st

        sex
survived female male Total
   no         9  120   129
   yes      134   59   193
   Total    143  179   322

, , passengerClass = 2nd

        sex
survived female male Total
   no        13  148   161
   yes       94   25   119
   Total    107  173   280

, , passengerClass = 3rd

        sex
survived female male Total
   no       134  440   574
   yes       79   58   137
   Total    213  498   711

, , passengerClass = Total

        sex
survived female male Total
   no       156  708   864
   yes      307  142   449
   Total    463  850  1313
```

```
tab(titanic, ~ survived + sex + passengerClass)  %>%
  barchart(auto.key=T)
```



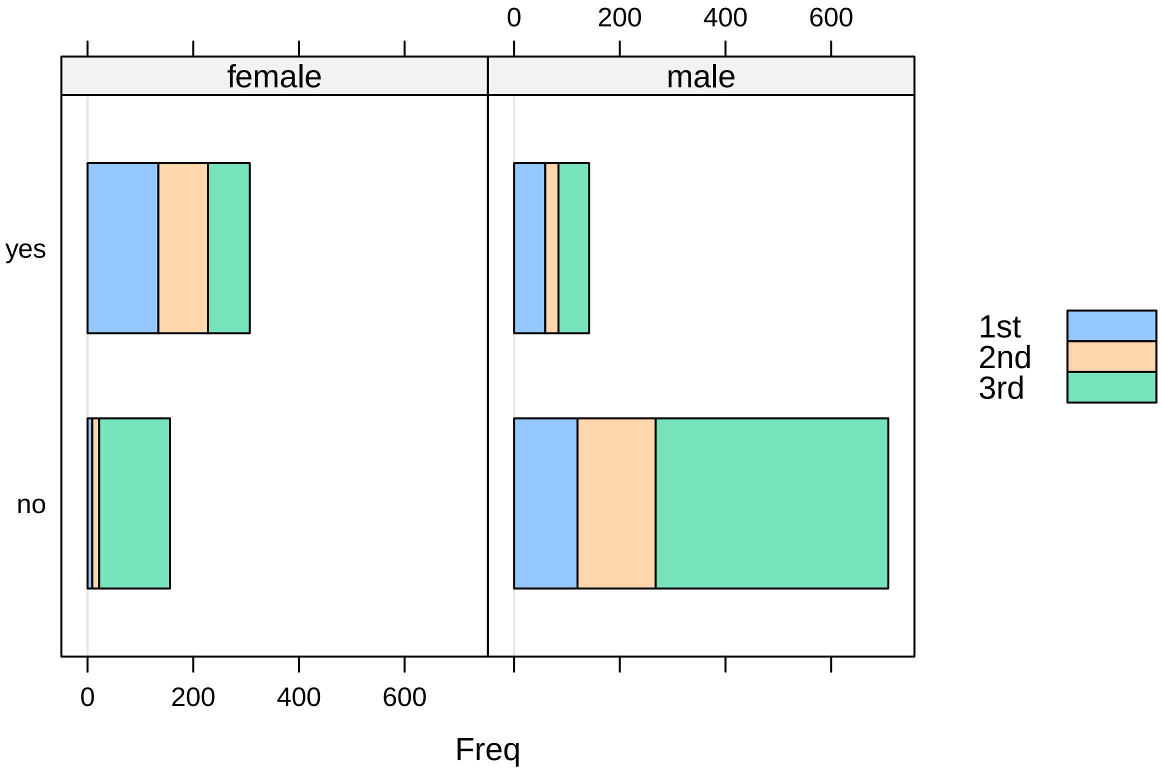
Get rid of 'Total'

Using `tab_` instead of `tab` suppresses the "Total" margin which we don't want to display since it is redundant in the graph. Using `tab__` suppresses both the "Total" margin the the "All" margin in conditional tables generated by using the arguments `pct` or `pr`.

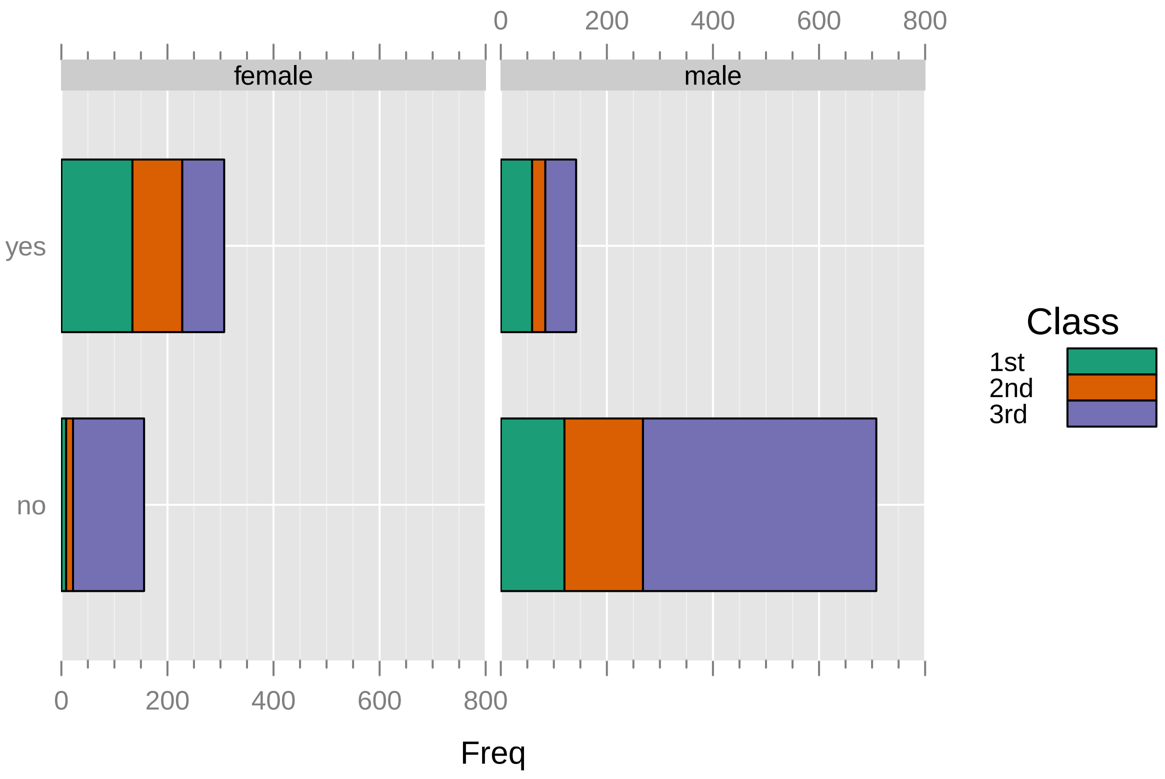
```
tab_(titanic, ~ survived + sex + passengerClass)
```

```
, , passengerClass = 1st

        sex
survived female male
     no        9  120
     yes     134   59

, , passengerClass = 2nd

        sex
survived female male
     no       13  148
     yes      94   25

, , passengerClass = 3rd

        sex
survived female male
     no      134  440
     yes      79   58
```

```
tab_(titanic, ~ survived + sex + passengerClass)  %>%
  barchart(auto.key=T)
```



```
gd() # ggplot2-like appearance
tab_(titanic, ~ survived + sex + passengerClass)  %>%
  barchart(
    auto.key = list(space = 'right', title='Class'),
    xlim = c(0, 800))
```



This is not really informative. We want to see the relative proportion of survivors in the different subgroups so we change the order of the variables. Also the labels 'yes' and 'no' are not informative themselves.

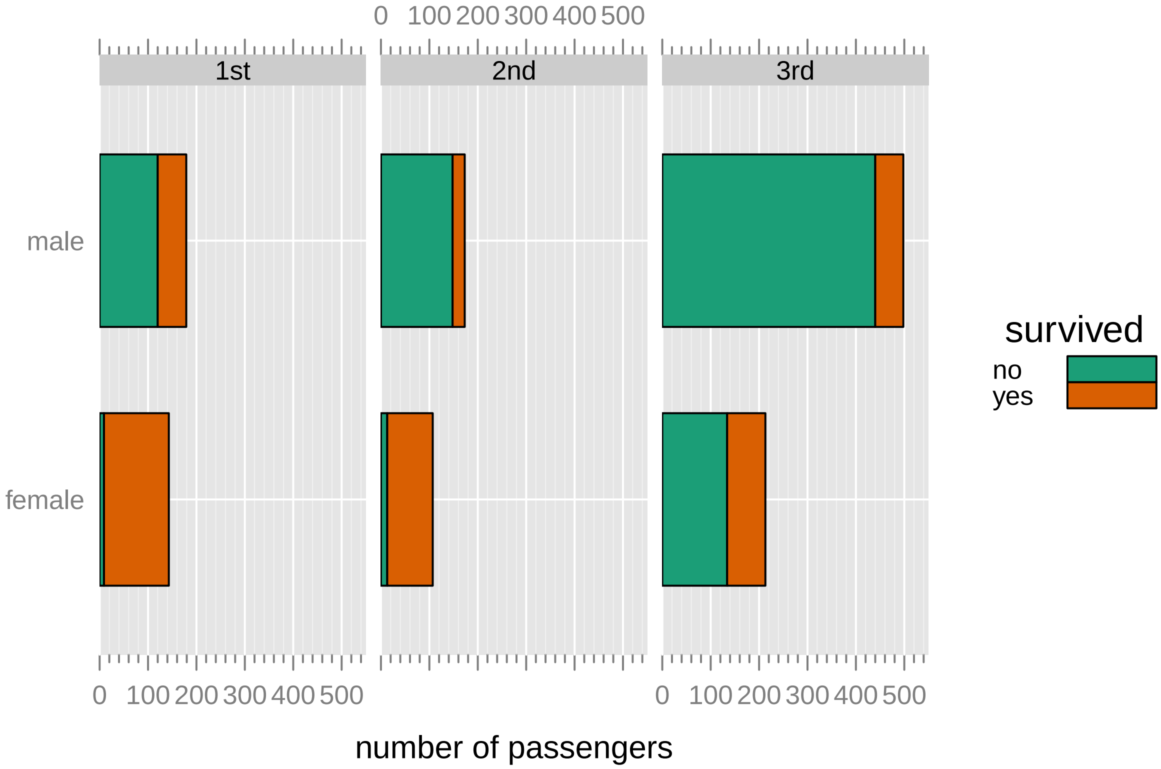
We change the order of the variables in the formula so the variable we want to see within bars comes last.

The first variable generates different bars within panels and the second variable generates the panels.

```
tab_(titanic, ~  sex + passengerClass + survived)
```

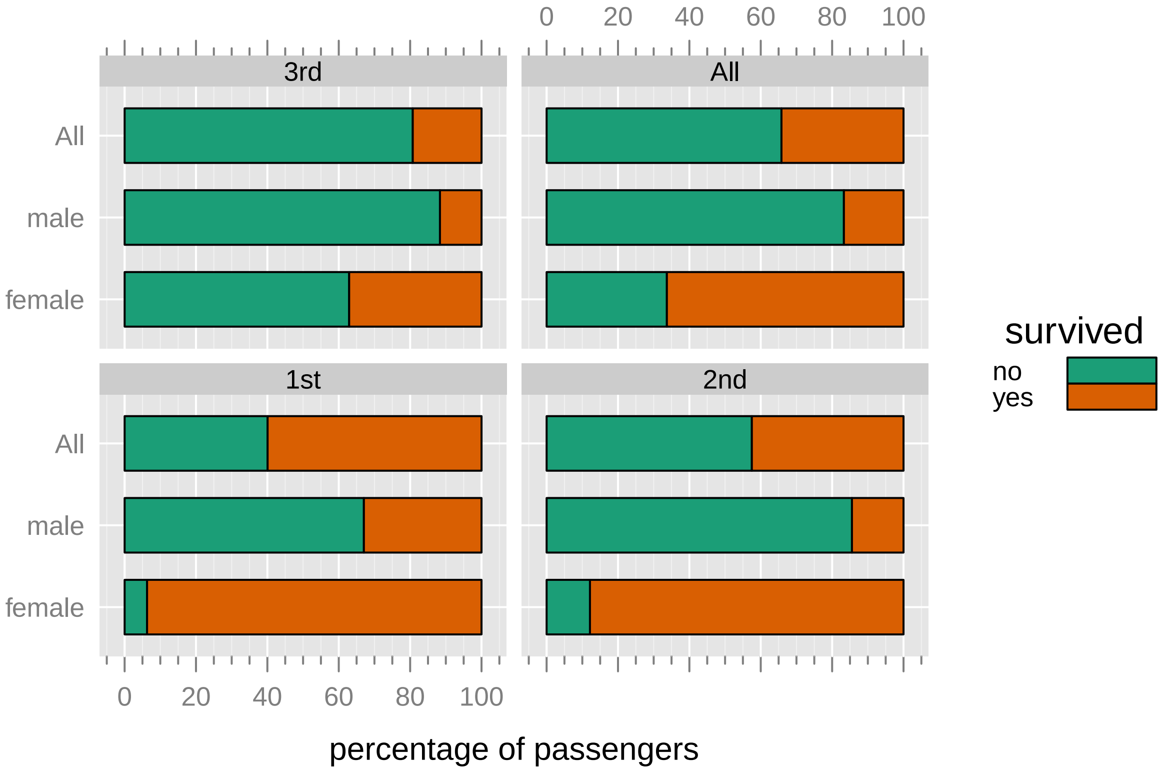
```
, , survived = no

        passengerClass
sex      1st 2nd 3rd
  female   9  13 134
  male   120 148 440

, , survived = yes

        passengerClass
sex      1st 2nd 3rd
  female 134  94  79
  male    59  25  58
```

```
tab_(titanic, ~  sex + passengerClass + survived)  %>%
  barchart(xlab = 'number of passengers',
           xlim = c(0,550),
           auto.key=list(space='right',title='survived'))
```



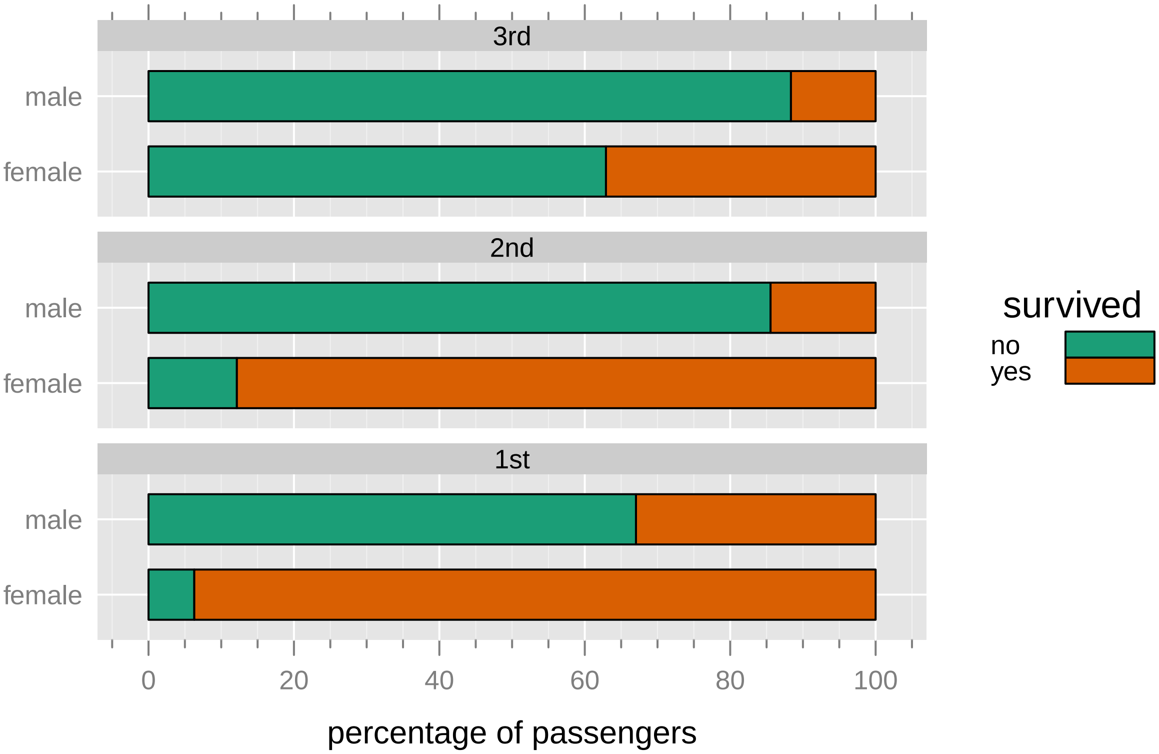
But if we want to emphasize proportions we really want proportions

```
tab_(titanic, ~  sex + passengerClass + survived, pct = c(1,2)) %>%
  barchart(xlab = 'percentage of passengers',
           auto.key=list(space='right',title='survived'))
```



This shows overall proportions as well as within-gender proportions. To get rid of both 'All' and 'Total'. Use `tab__` instead of `tab`.

```
tab__(titanic, ~  sex + passengerClass + survived, pct = c(1,2)) %>%
  barchart(xlab = 'percentage of passengers', layout = c(1,3),
           auto.key=list(space='right',title='survived'))
```

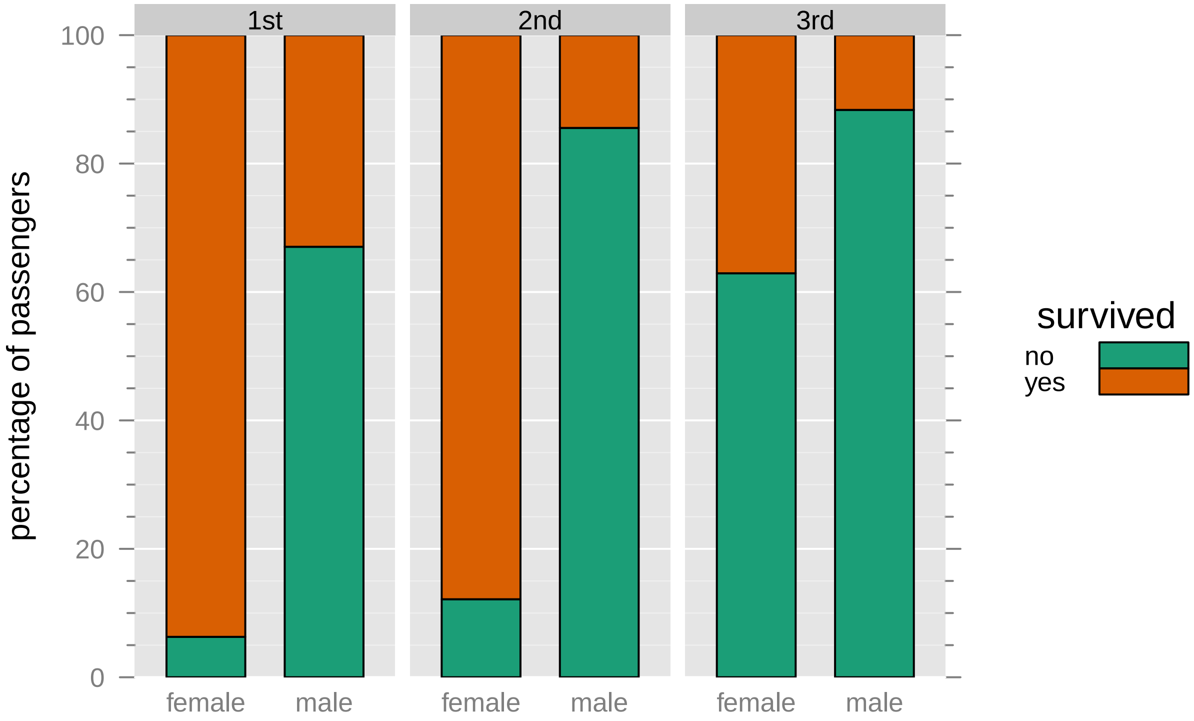


experimenting:

```
tab__(titanic, ~  sex + passengerClass + survived, pct = c(1,2))
```

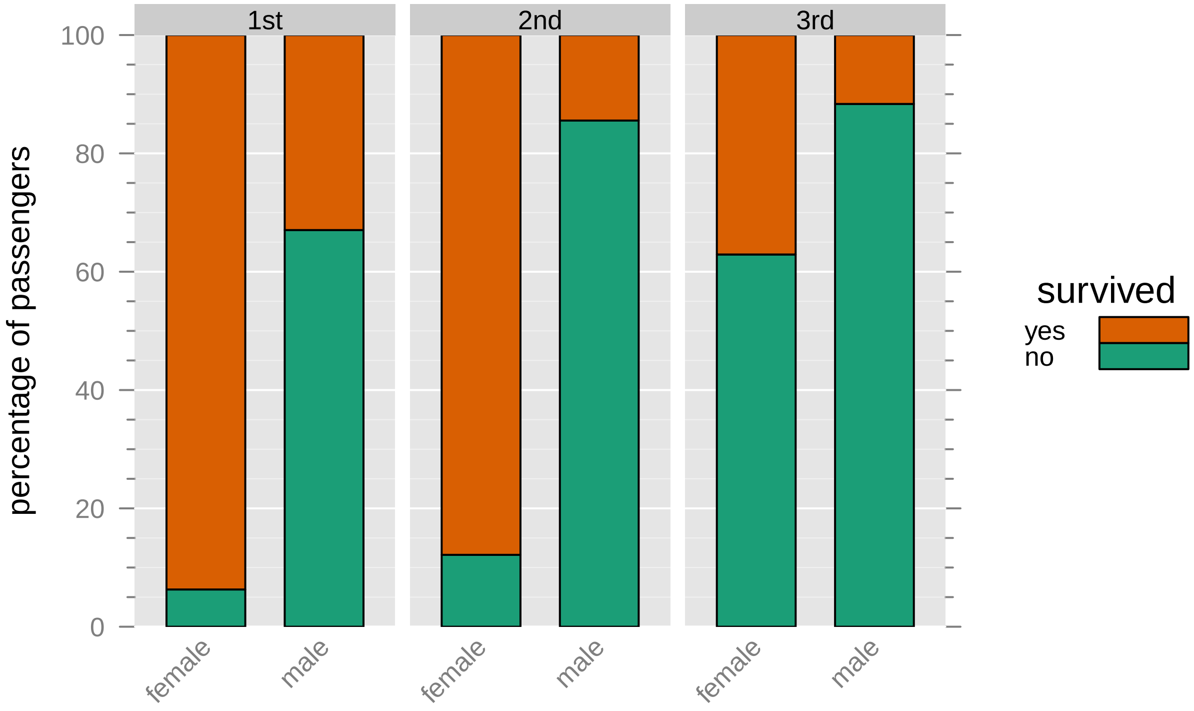
```
, , survived = no

        passengerClass
sex            1st       2nd       3rd
  female  6.293706 12.149533 62.910798
  male   67.039106 85.549133 88.353414

, , survived = yes

        passengerClass
sex            1st       2nd       3rd
  female 93.706294 87.850467 37.089202
  male   32.960894 14.450867 11.646586
```

```
tab__(titanic, ~  sex + passengerClass + survived, pct = c(1,2)) %>%
  barchart(ylab = 'percentage of passengers',
           horizontal = FALSE,
           ylim = c(0,100), layout = c(3,1),
           auto.key=list(space='right',title='survived'))
```



Looks wrong because order of colors are different in graph and in legend

```
tab__(titanic, ~  sex + passengerClass + survived, pct = c(1,2)) %>%
  barchart(ylab = 'percentage of passengers',
           horizontal = FALSE,
           ylim = c(0,100), layout = c(3,1),
           scales = list(x=list(rot=45)),
           auto.key=list(space='right',title='survived', reverse.rows = T))
```

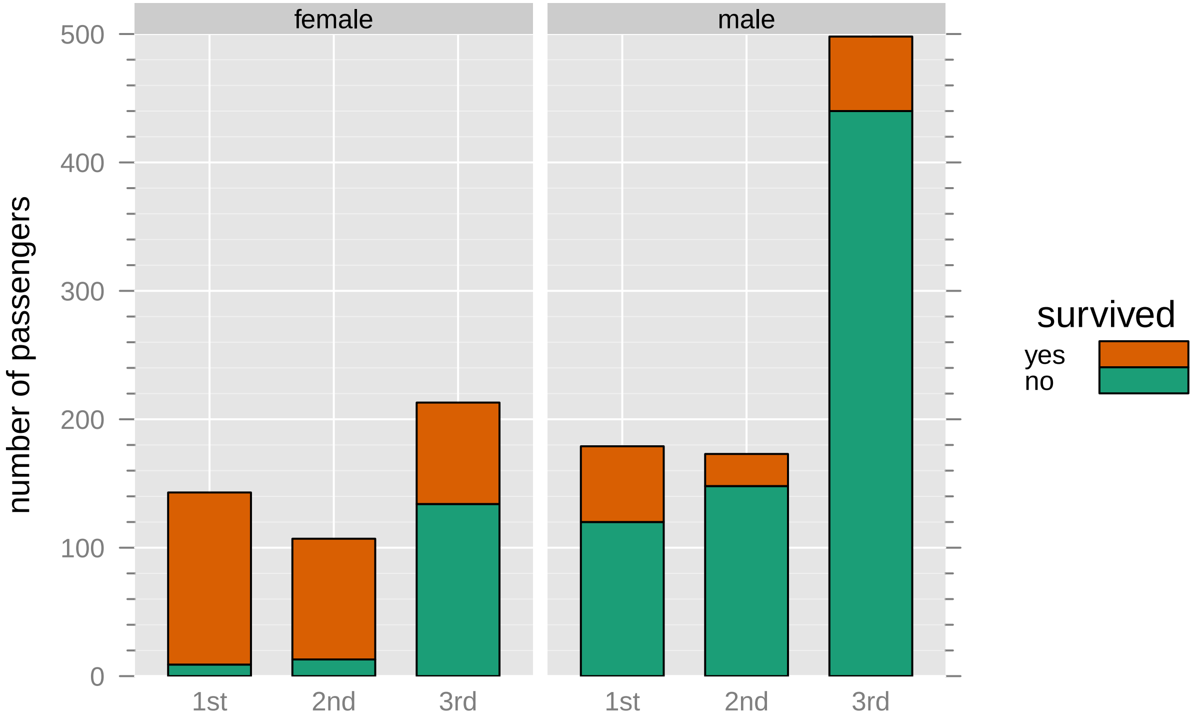


more experimenting:

```
tab__(titanic, ~  passengerClass + sex + survived)
```

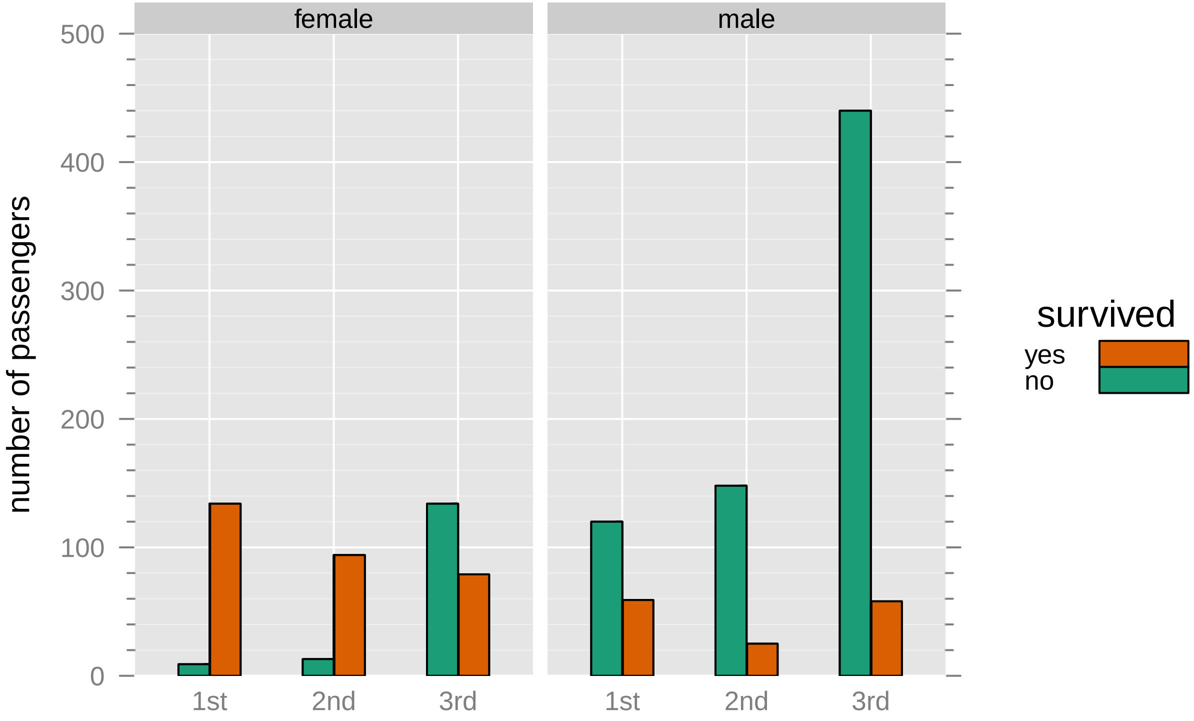
```
, , survived = no

              sex
passengerClass female male
           1st      9  120
           2nd     13  148
           3rd    134  440

, , survived = yes

              sex
passengerClass female male
           1st    134   59
           2nd     94   25
           3rd     79   58
```

```
gd()
tab__(titanic, ~  passengerClass + sex + survived) %>%
  barchart(ylab = 'number of passengers',
           ylim = c(0,500),
           horizontal = FALSE,
           auto.key=list(space='right',title='survived',
                      reverse.rows=T))
```



Note that ‘reverse.rows’ get the key to show colors in the same order as in the plot.

```
tab__(titanic, ~  passengerClass + sex + survived) %>%
  barchart(ylab = 'number of passengers',
           ylim = c(0,500),
         #  box.width = rep(.9^c(2,1,1,1,1,1), each = 2),
           box.width = c(1,5,5,.1)/10,
         box.ratio = 2, stack = F,
           horizontal = FALSE,
           auto.key=list(space='right',title='survived', reverse.rows = T))
```



Questions

- What are the pros and cons of these various graphs?
- Can you find good visualizations of the Titanic data?
- What factors make a visualization effective?
- You’ve heard “a picture is worth a thousand words”. But how about: “a picture is worthless unless it tells a good story.” Do any of these plots tell a good story?
- What factors detract from the effectiveness of a visualization?
- Can you think of ways of presenting this data that would be more effective than the ones above?
- If you are familiar with ‘ggplot2’, use it to present this data?
- Try mosaic plots in the ‘vcdExtra’ package.