First Steps to R Packages

W. John Braun, UBC

Workshop - SSC Student Conference

May 28, 2022

First Steps to Writing a Package

We will go through the basic steps of constructing a package, via an example where 4 functions and 1 data set are to be combined into a single package and prepared for submission to CRAN.

The functions and data set will relate to the two-component mixture distribution having density function

$$f(x) = pf_1(x) + (1-p)f_2(x)$$

where $p \in (0,1)$**,** $f_1(x)$ **and** $f_2(x)$ **are pdfs of non-central** t **random variables.**

The t **random variables are governed by a non-centrality parameter** `ncp` **which locates their center and by the number of degrees of freedom** `df`**.**

The package functions

The functions that will make up the package are designed to

- **simulate the random variables (`rtmix()`)**
- **calculate quantiles (`qtmix()`)**
- **calculate probability densities (`dtmix()`) and**
- **calculate cumulative probabilities (`ptmix()`)**

The functions

The random variate generator: `rtmix()`

```
rtmix(5, df = c(4, 7), ncp = c(-1, 5), PI = .24)

## [1] -0.7637152  3.8608394  5.7320305  7.2167542  5.8635216
```

5 randomly generated values from the mixture distribution

- **with 4 and 7 degrees of freedom**
- **with component means of -1 and 5**
- **where the first component has probability 0.24, i.e.** $p = 0.24$

The functions

The quantile function: `qtmix()`

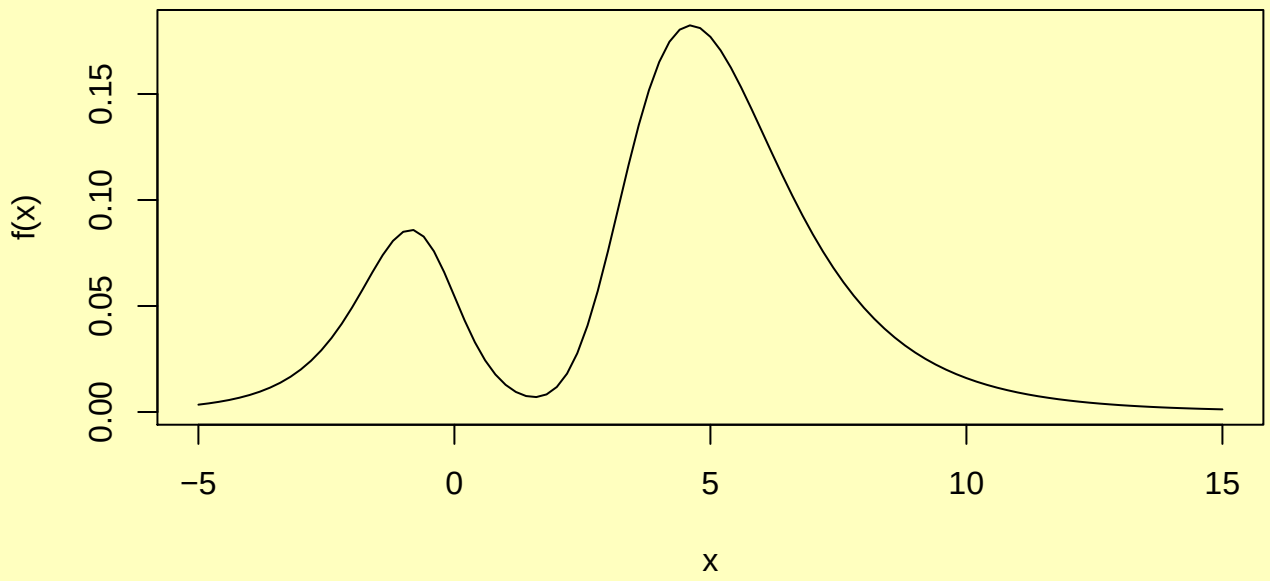
```
qtmix(c(.25, .75, .9), df = c(4, 7), ncp = c(-1, 5),
      PI = .24)

## [1] 2.453922 6.096464 7.806849
```

The 25th, 75th and 90th percentiles of the mixture distribution.

The functions

The probability density function calculator: `dtmix()`

```
curve(dtmix(x, df = c(4, 7), ncp = c(-1, 5), PI = .24),
    from = -5, to = 15, ylab = "f(x)")
```

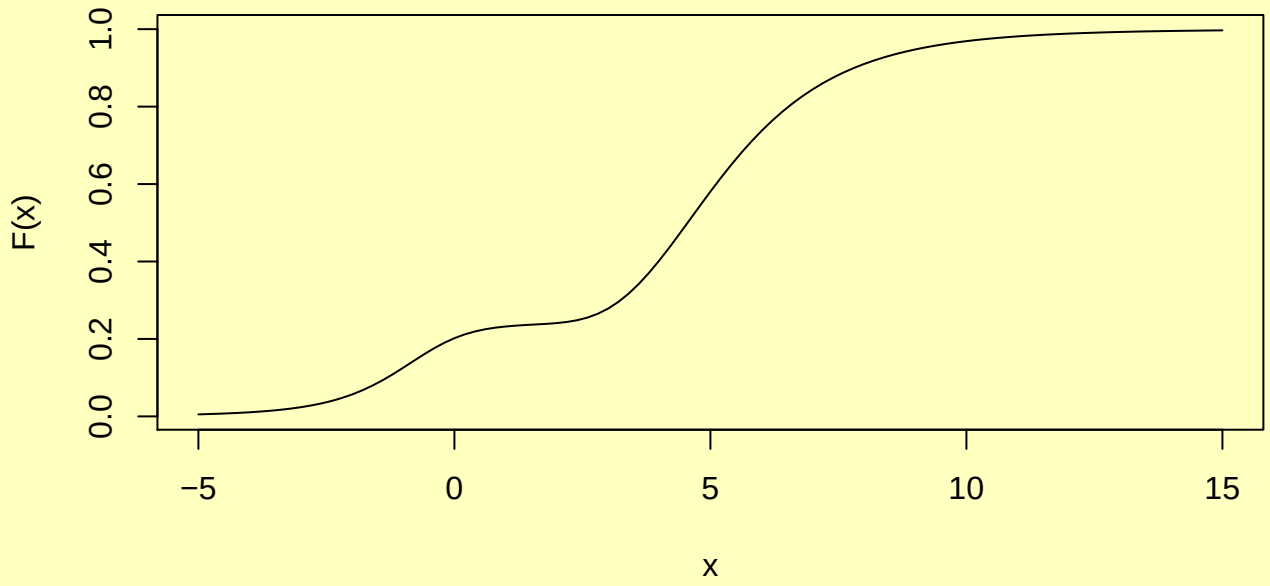


A plot of the probability density curve of the mixture distribution.

The functions

The cumulative distribution function calculator: `ptmix()`

```
curve(ptmix(x, df = c(4, 7), ncp = c(-1, 5), PI = .24),
    from = -5, to = 15, ylab = "F(x)")
```



A plot of the cdf curve for the mixture distribution.

The data

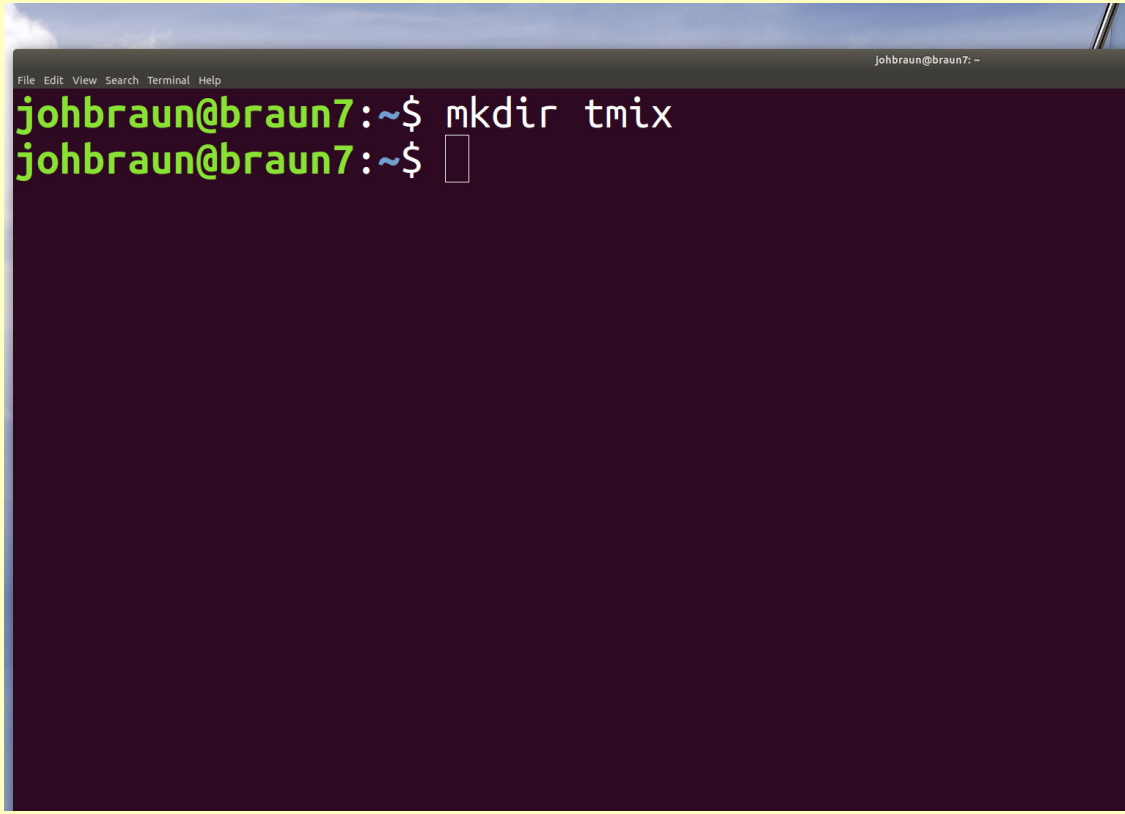
The data file is called `simdata`:

```
str(simdata)


## 'data.frame': 200 obs. of  2 variables:
##  $ x: num  -0.043 -1.211 -1.079 -0.383 -1.381 ...
##  $ y: num  -0.748 -2.713 -2.518 -3.78 -2.62 ...
```

Building the package directory

Since the functions concern a 2-component mixture of t random variables, we name the package *tmix*.

Our first step is to create a package directory which has that name.



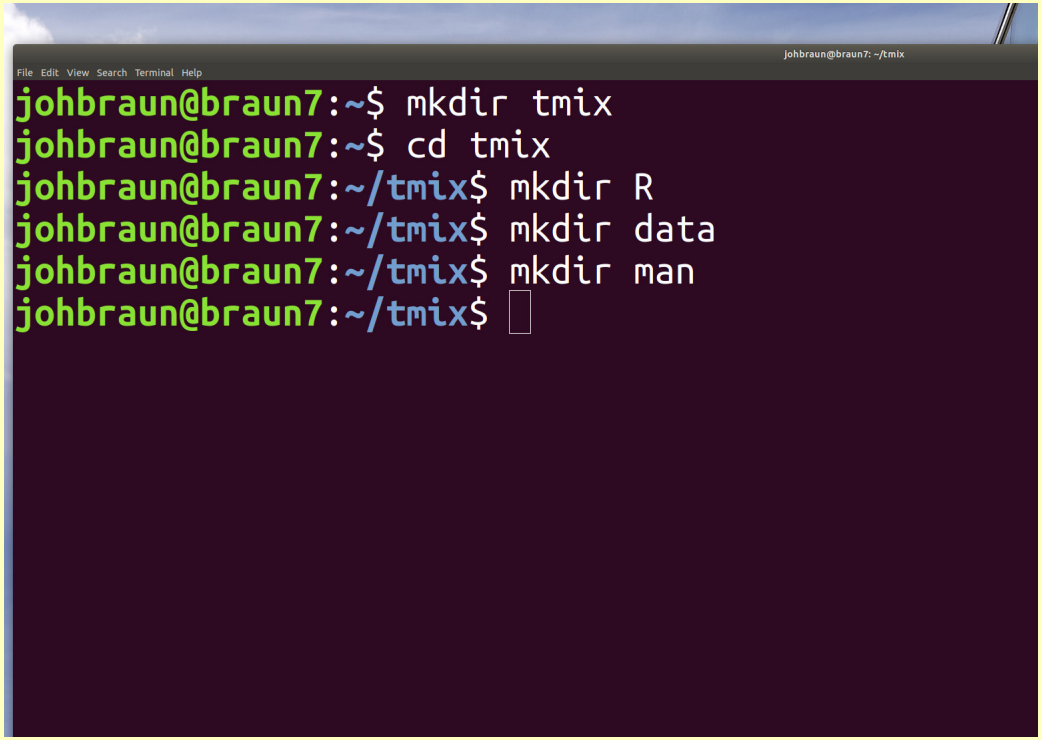
Building the package directory

Within the *tmix* directory, we require the following:

- A *man* directory.

We need at least one, usually both, of the following

- An *R* directory.
- A *data* directory.



Building the package directory

Within the *tmix* directory, we also minimally require the following text files:

- **A *DESCRIPTION file*.**
- **A *NAMESPACE file*.**

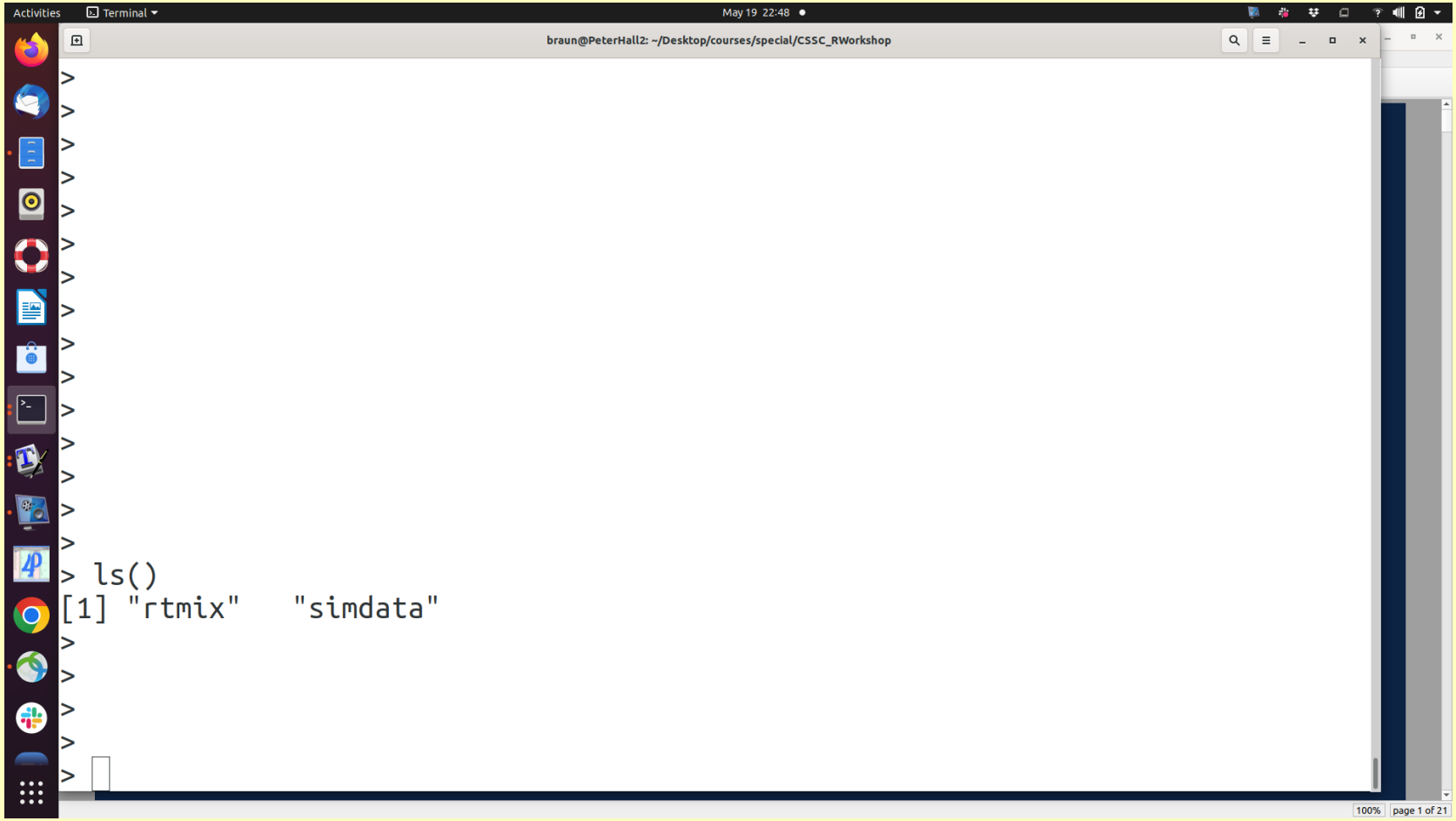
Building the package directory - automatic method

You can also construct the package directories from within R or RStudio using the `package.skeleton` function. This function is built into the default R distribution.

You need at least one function or data object in your workspace in order to use this function.

Start with only one or two objects and complete the build and check process on a package that only contains them. Then gradually add functions and data to your package, rebuilding and rechecking at each stage. This will make debugging easier and faster.

Building the package directory - automatic method



Here, I am starting with one function and one data object in the workspace.

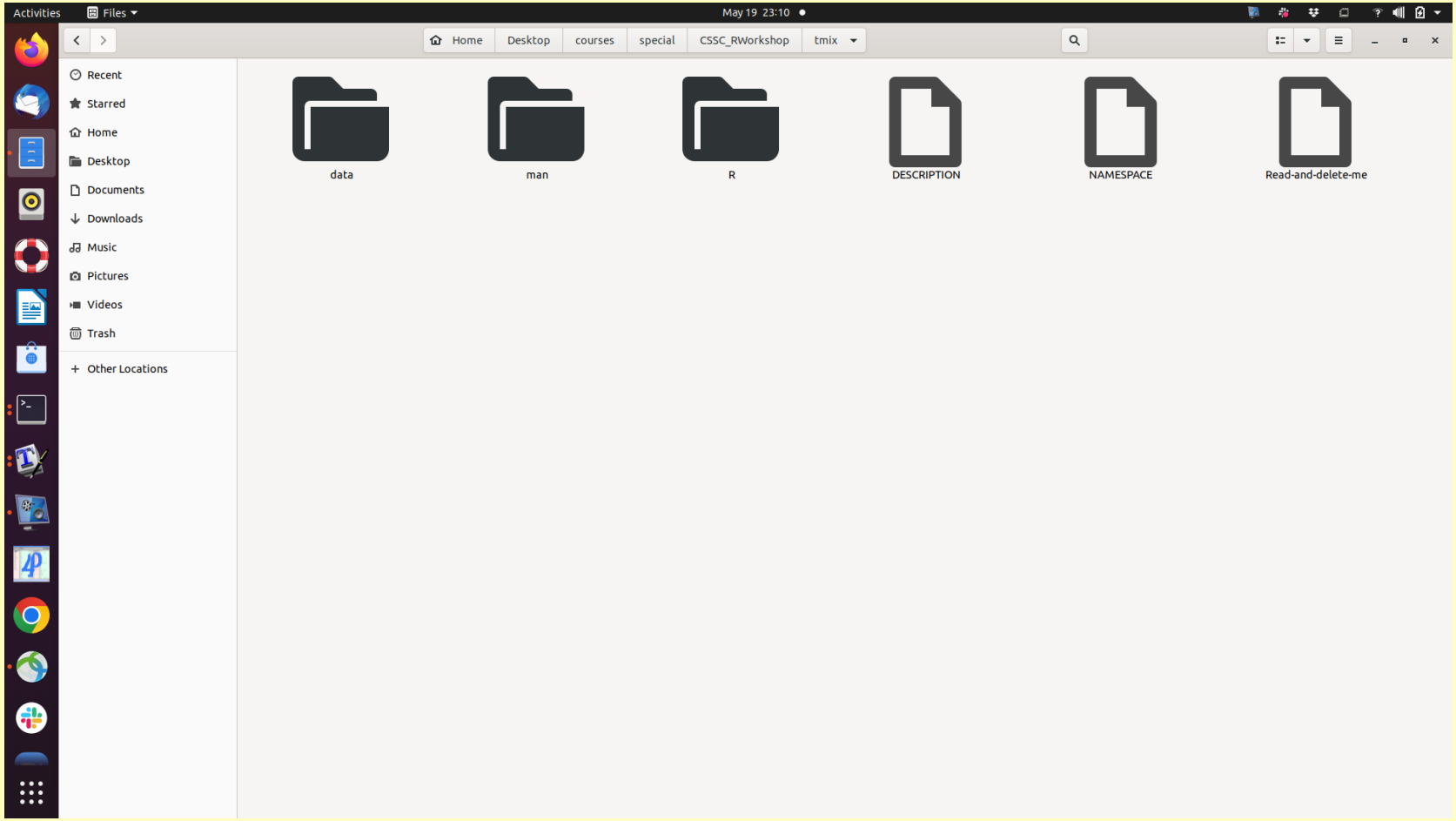
Building the package directory - automatic method

```
> ls()
[1] "rtmix"   "simdata"
> package.skeleton("tmix")
Creating directories ...
Creating DESCRIPTION ...
Creating NAMESPACE ...
Creating Read-and-delete-me ...
Saving functions and data ...
Making help files ...
Done.
Further steps are described in './tmix/Read-and-delete-me'.
>
```

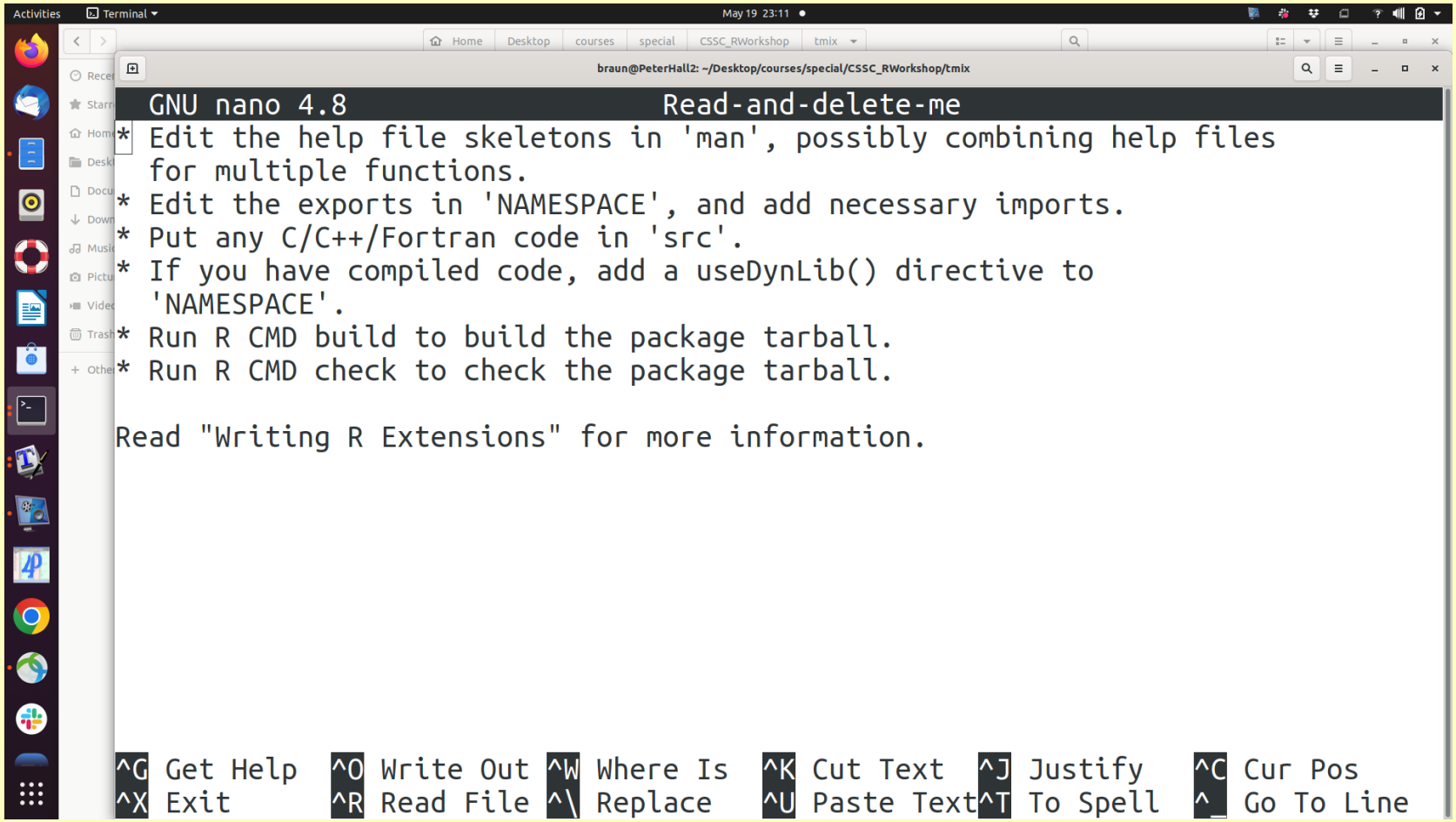
The command `package.skeleton("tmix")` creates the directory structure with the beginnings of the files but just for the two objects. I need to add in the other three objects, one at a time, later.*

*Watch the award-winning 19-second silent movie packageSkeleton.mp4 to see the commands in action.

Building the package directory - automatic method

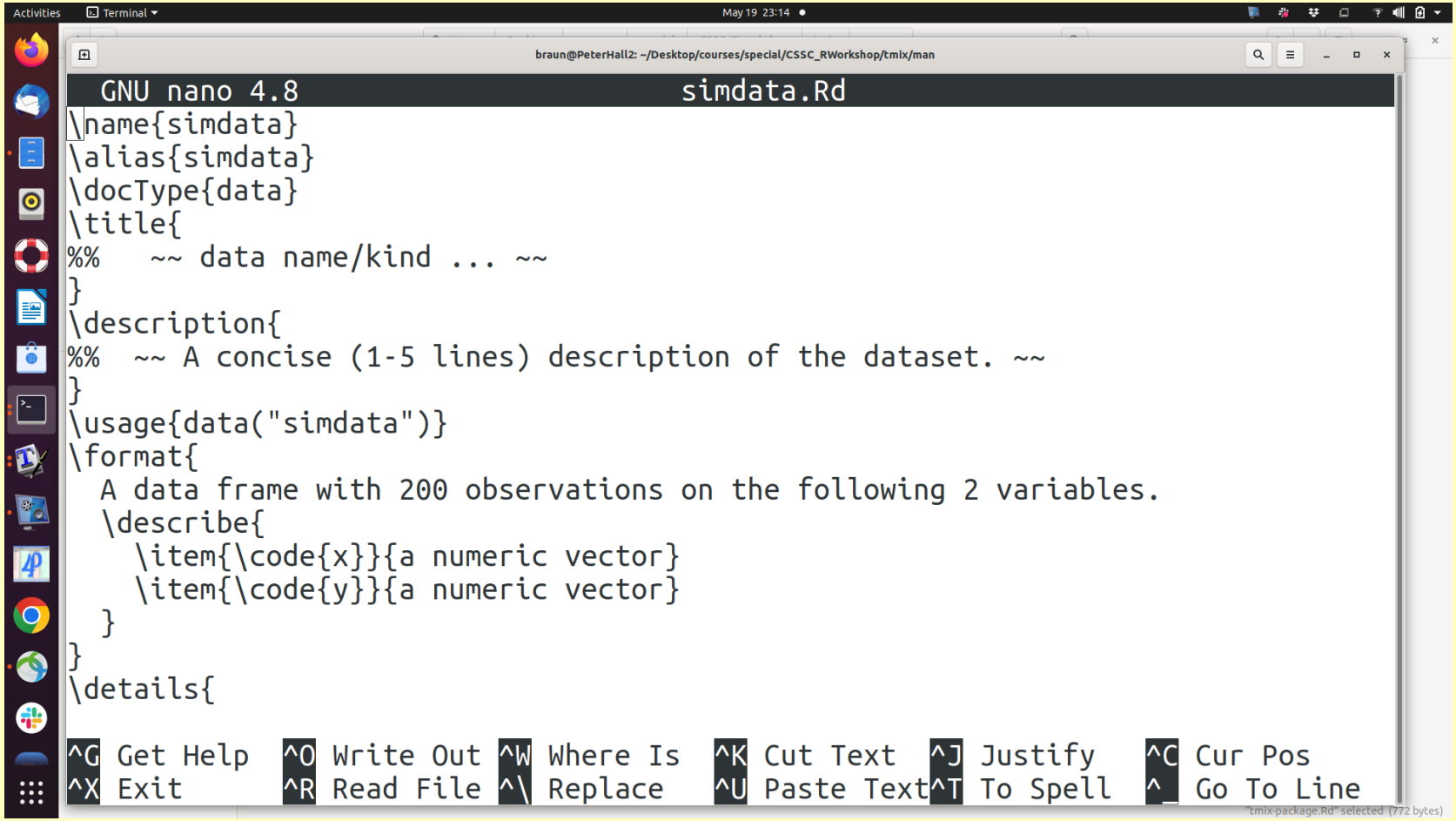


Building the package directory - automatic method

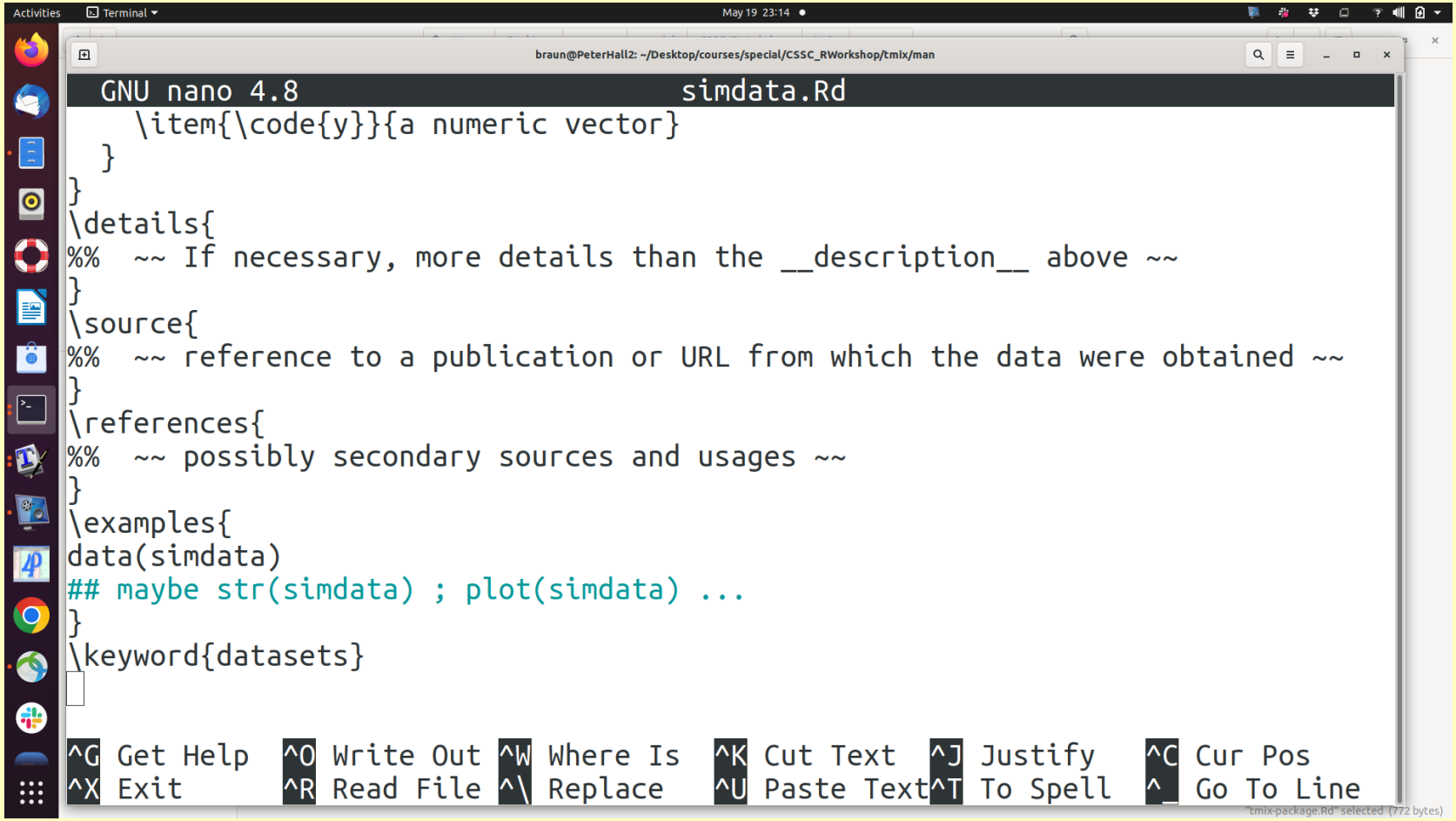


The third and fourth * points can be ignored unless you are doing something more sophisticated. Writing R Extensions can be found on the CRAN site. You do need to consult this.

Building the package directory - automatic method

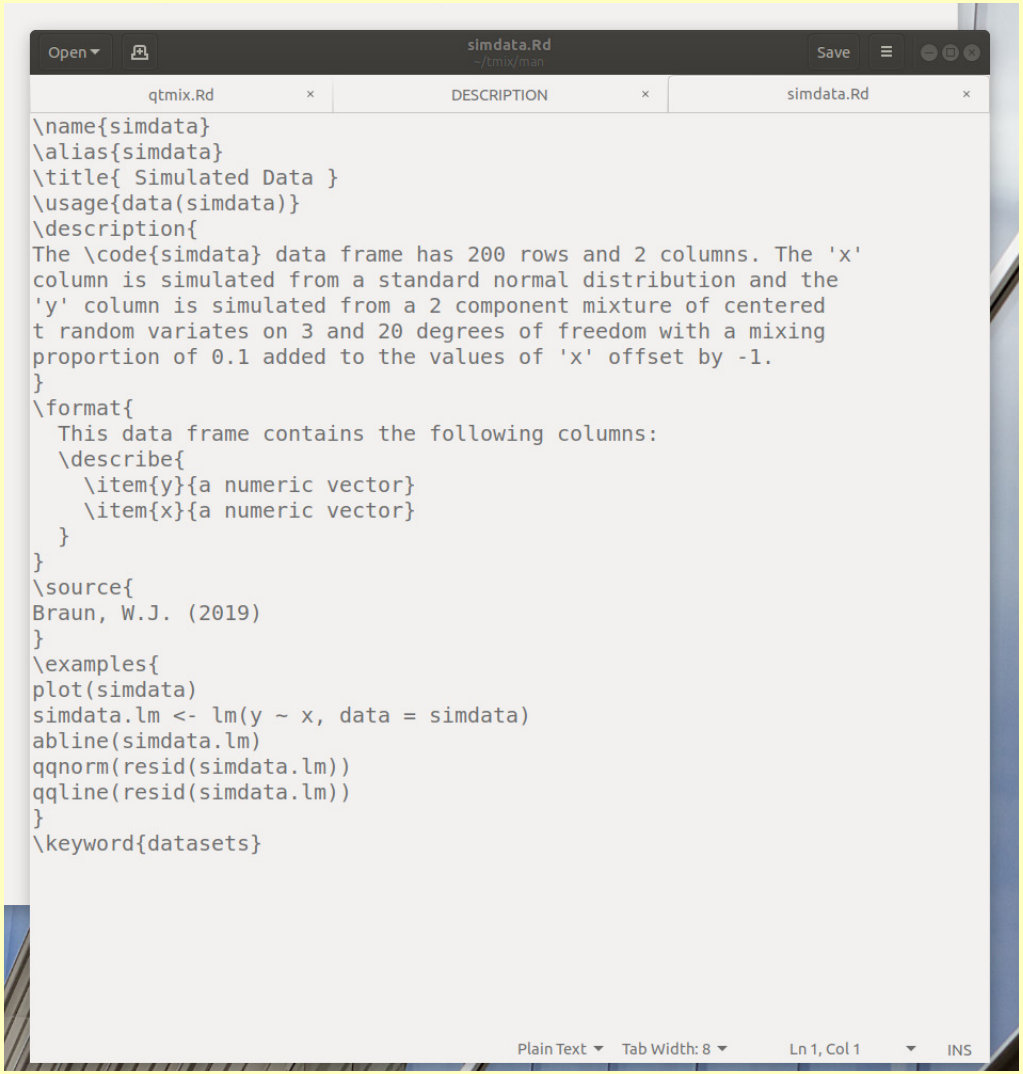


Building the package directory - automatic method



Cleaning up the man file - automatic method

The contents of *simdata.Rd* should be changed to:

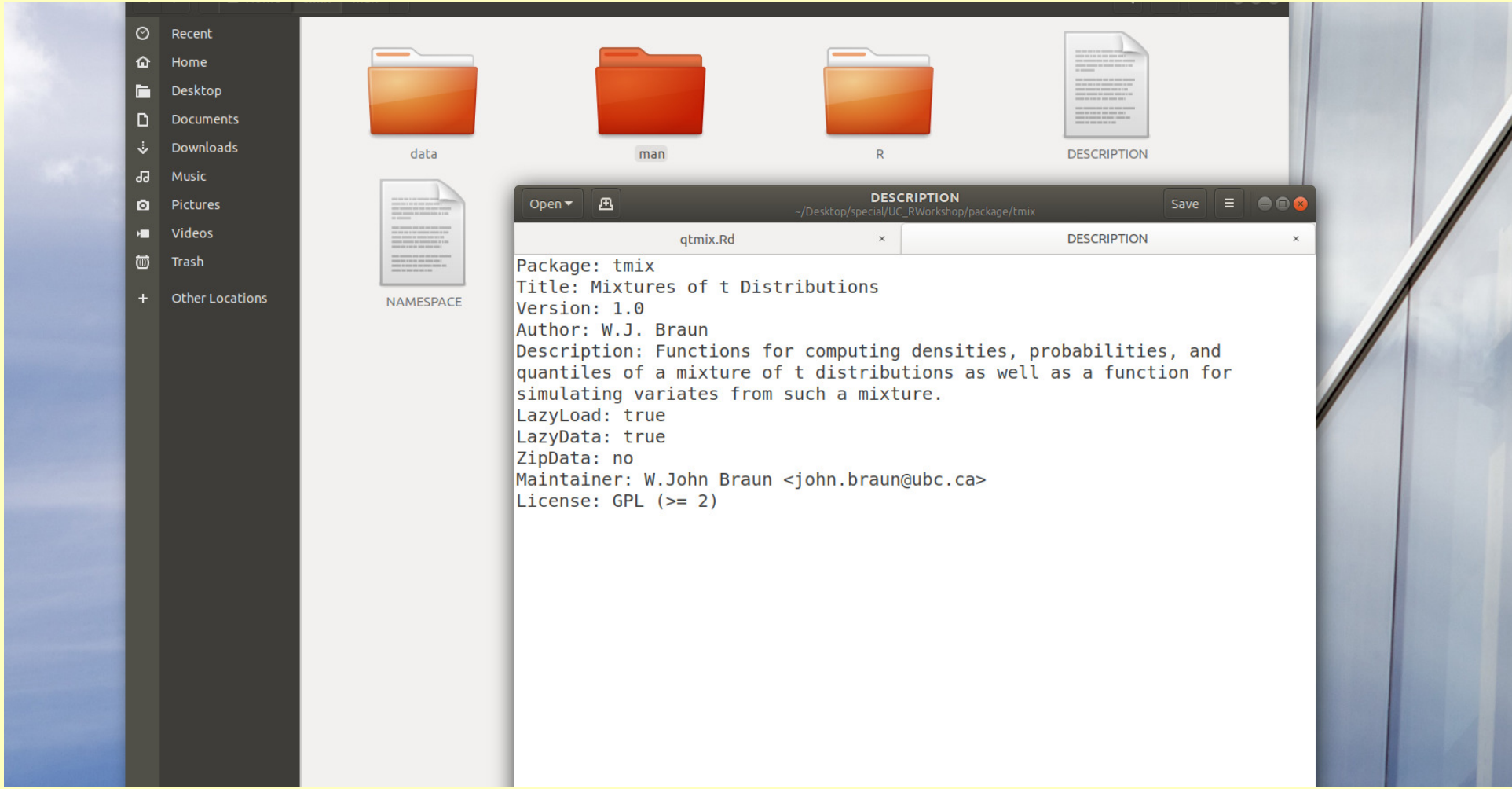



```
\name{simdata}
\alias{simdata}
\title{ Simulated Data }
\usage{data(simdata)}
\description{
The \code{simdata} data frame has 200 rows and 2 columns. The 'x'
column is simulated from a standard normal distribution and the
'y' column is simulated from a 2 component mixture of centered
t random variates on 3 and 20 degrees of freedom with a mixing
proportion of 0.1 added to the values of 'x' offset by -1.
}
\format{
  This data frame contains the following columns:
  \describe{
    \item{y}{a numeric vector}
    \item{x}{a numeric vector}
  }
}
\source{
Braun, W.J. (2019)
}
\examples{
plot(simdata)
simdata.lm <- lm(y ~ x, data = simdata)
abline(simdata.lm)
qqnorm(resid(simdata.lm))
qqline(resid(simdata.lm))
}
\keyword{datasets}
```

But we're getting ahead of ourselves.

The DESCRIPTION file

Possible contents for the DESCRIPTION are as follows:



The contents of the `Description` should all be on one line. Multiple lines are used here for display purposes only.

The *NAMESPACE file*

This file is needed so that you don't have to worry that the names you choose for your functions or data sets will conflict with names of functions and data sets from other packages.

The contents of NAMESPACE for this example:

```
importFrom("stats", "qt", "dt", "pt", "rt", "rbinom")
exportPattern(".")
export("dtmix", "ptmix", "qtmix", "rtmix")
```

We have not included our data file in the export list, only the functions.

We have included all 4 of our functions in the export list.

If we had held one back, the package user would not have direct access to that function.

The R and data directories

We create four files in the *R* subdirectory, calling them

- ***qtmix.R***
- ***rtmix.R***
- ***dtmix.R***
- ***ptmix.R***

The contents of these files are the corresponding functions.

The data, stored in a file named *simdata.R* will be copied into the *data* subdirectory.

The help directory: man

The *man* directory will contain 5 files:

- ***qtmix.Rd***
- ***rtmix.Rd***
- ***dtmix.Rd***
- ***ptmix.Rd***
- ***simdata.Rd***

Here, we will construct those files manually.
Optionally, you can include a help file for the package itself called *tmix-package.Rd*. This gives an overview of the package.

You can avoid some of this work by using the *roxygen2* package (Wickham et al, 2019).

The help directory: man

The contents of *qtmix.Rd* are:

qtmix.Rd
~/Desktop/special/UC_RWorkshop/package/tmix/man

```
\name{qtmix}
\alias{qtmix}
\title{Quantile of Mixture of t Distributions}
\description{
Quantile function for a 2-component mixture of t distributions
with 'df' degrees of freedom and non-central parameter 'ncp'.
}
\usage{qtmix(p, df, ncp, PI, lower.tail = TRUE, tol = 1e-7)}
\arguments{
  \item{p}{a vector of probabilities.}
  \item{df}{a vector (length 2) of degrees of freedom.}
  \item{ncp}{a vector (length 2) of noncentrality parameters.}
  \item{PI}{a numeric constant giving the mixture parameter.}
  \item{lower.tail}{a logical constant which is TRUE if the lower tail
quantiles are required.}
  \item{tol}{a control parameter for the accuracy imposed on the quantile
calculation.}
}
\value{A vector of quantiles.}
\details{The quantiles are obtained by employing a Newton-Raphson
iteration to solve the inverse problem.}
\references{
Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) _Continuous
     Univariate Distributions_, volume 2.  Wiley,
     New York.
}
\seealso{\code{\link{qt}}}
\author{W.J. Braun}
\examples{
     qtmix(c(.1, .3, .6), df=c(2, 7), ncp=c(-4, 0.5), PI = 0.75)
}
\keyword{distribution}
```

Plain Text Tab Width: 8 Ln 9, Col 12 INS

The help directory: man

Most of the contents of `qtmix.Rd` are required, with the exception of the following: `details`, `seealso`, `references`, `author` and `examples`.

These options are strongly recommended, since the more detail that is provided, the better.

For a list of keywords that you might use, try

```
RShowDoc("KEYWORDS")
```

Other pieces of a package

The `qtmix()` function invoked a Newton-Raphson iteration to solve the required inverse problem.

Since this iteration requires a loop, it may have been better to write the code into a *C* or *Fortran* program that would be called from R.

If this had been done, the external code would be written in the form of a subroutine or collection of subroutines and stored in an additional directory called *src*.

Special commands are required within the R functions (e.g. `qtmix()`) which would need to access the external code. The R manual has more information on this.

Documentation for the package itself is also recommended, as is the creation of vignettes, which describe the package in action.

Building, checking and submitting

Once the pieces of the package are assembled, it is necessary to build the package. In RStudio, this is fairly straightforward.*

At the command line (Mac, Linux or Cygwin in Windows), we would type

```
R CMD build tmix
```

To check that the package components are properly constructed, we next type

```
R CMD check tmix
```

It is usually advisable to build and check the package as it is being constructed, instead of waiting until all the pieces have been assembled.

*See the 99 second silent movie called RStudiobuildCheck.mp4 for checking and building a copy of the package called *tmix2*.

Building, checking and submitting

Before submitting the package to CRAN, we need to do a more thorough check as follows:

```
R CMD check --as-cran tmix
```

This last check is to be done using the most recent development version of R. Before submitting to CRAN, the submittor should check the R manual to ensure that all requirements have been met.

The final steps in the submission can be carried out at `https://cran.r-project.org/submit.html`.

Alternatively, and maybe preferably, you could post your package on a github site to encourage collaborative development of it.