```r
## Load libraries
library(splines)
library(MASS)

## Define the number of tests
ntest <- 1000

## Set nuber of simulations
nSims <- 100##10000

##second shape parameter for beta distribution
shape2 <- 2

source("https://raw.githubusercontent.com/leekgroup/fdrreg/master/R/estPi0Reg.R")
```

Function to generate p-values:

```r
genPvals <- function(pi0, shape2)
{
  ntest <- length(pi0)

  nullI <- rbinom(ntest,prob=pi0,size=1)> 0

  pValues <- rep(NA,ntest)
  pValues[nullI] <- runif(sum(nullI))
  pValues[!nullI] <- rbeta(sum(!nullI),1,shape2)

  pValues
}
```

1 Probability of being a false positive as a linear function of time

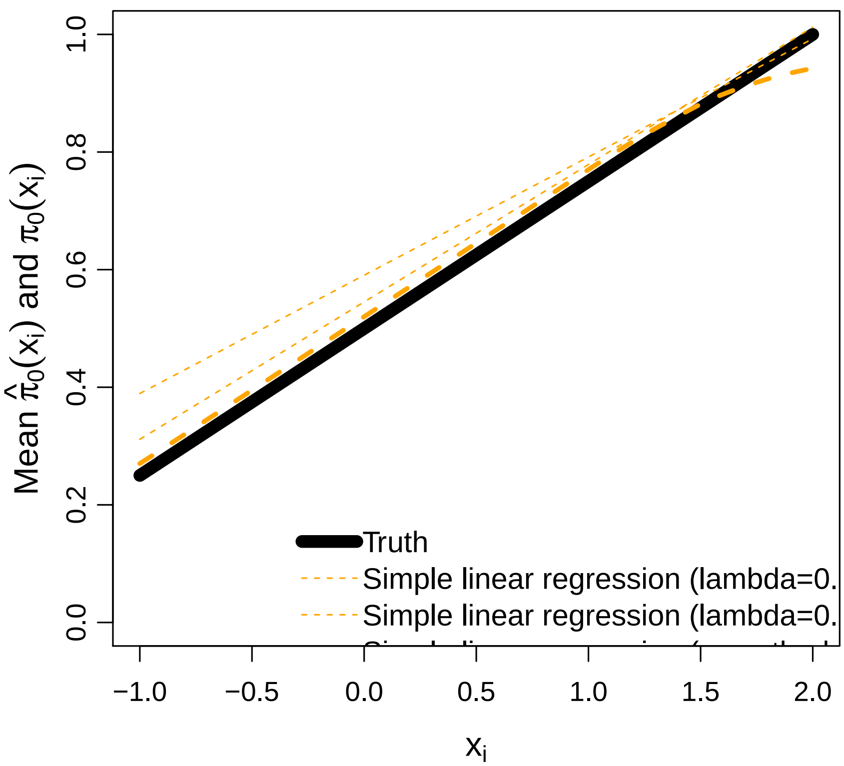
```r
set.seed(1345)

## Set up the time vector and the probability of being null
tme <- seq(-1,2,length=ntest)
pi0 <- 1/4*tme+1/2

##save the value of pi0hat for each simulation for lambda=0.8, 0.9, final smoothed value
pi0hatMat0.8 <- pi0hatMat0.9 <- pi0hatMatFinal <-
  matrix(NA, nrow=nSims, ncol=ntest)
```

```r
for(sim in 1:nSims)
  {

  if(sim%%100 == 0)
  {
    print(sim)
  }

  ##generate p-values
  pValues <- genPvals(pi0, shape2)

  ## Get the estimate
  pi0Est <- estPi0Reg(pValues, X=tme, smooth.df=3)

  pi0hatMat0.8[sim, ] <- pi0Est$pi0.lambda[,round(pi0Est$lambda,2)==0.8]
  pi0hatMat0.9[sim, ] <- pi0Est$pi0.lambda[,round(pi0Est$lambda,2)==0.9]
  pi0hatMatFinal[sim, ] <- pi0Est$pi0
}

## [1] 100

## Get the mean values:
pi0hatMean0.8 <- colMeans(pi0hatMat0.8)
pi0hatMean0.9 <- colMeans(pi0hatMat0.9)
pi0hatMeanFinal <- colMeans(pi0hatMatFinal)

##Get the variances:
pi0hatVar0.8 <- apply(pi0hatMat0.8, 2, var)
pi0hatVar0.9 <- apply(pi0hatMat0.9, 2, var)
pi0hatVarFinal <- apply(pi0hatMatFinal, 2, var)

##Get the variance bounds:
# zMat <- tmeInt
# S <- zMat%*%solve(t(zMat)%*%zMat)%*%t(zMat)
# pi0hatVarBound <-
#   diag(S)/(4*(1-0.8)^2)
```

1.1 Plot for means

```r
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0 ~ tme,col="black",type="l",lwd=8, lty=1,
     xlab="", yaxt = "n",
```

```
     ylim=c(0,1), ylab="")
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Mean ", hat(pi)[0](x[i])," and ", pi[0](x[i]))), 2, line=2, cex=1.3)
points(pi0hatMean0.8 ~ tme,col="orange",type="l",lwd=1, lty=2)
points(pi0hatMean0.9 ~ tme,col="orange",type="l",lwd=1, lty=2)
points(pi0hatMeanFinal ~ tme,col="orange",type="l",lwd=3, lty=2)

legend(x=-0.4, y=0.2,
       legend=c("Truth", "Simple linear regression (lambda=0.8)", "Simple linear regression
       col=c("black", "orange", "orange", "orange"), bty="n",
       lwd=c(8,1,1,3), lty=c(1,2,2,2),
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
axis(side=2, at=(0:5)/5, mgp=c(3, 0.7, 0))
```



1.2 Plot for variances

```
# par(cex.axis = 1.1, cex.main=1.3)
#
# plot(pi0hatVarBound ~ tme, col="red", ylim=c(0, max(pi0hatVarBound)),
#      lwd=3, lty=3,
#      type="l",
#      xlab="", ylab="")
# mtext(expression(x[i]), 1, line=3, cex=1.3)
# mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i])
# points(pi0hatVar ~ tme, col="black", type="l", lwd=3, lty=1)
# legend("top", ##x=-0.4, y=0.2,
#        legend=c("Empirical variance", "Upper bound"),
#        col=c("black", "red"), bty="n",
#        lwd=c(3,3), lty=c(1,3),
#        cex=1.2, x.intersp=0.2, y.intersp=1.0)
```

2 Probability of being a false positive as a smooth function of time

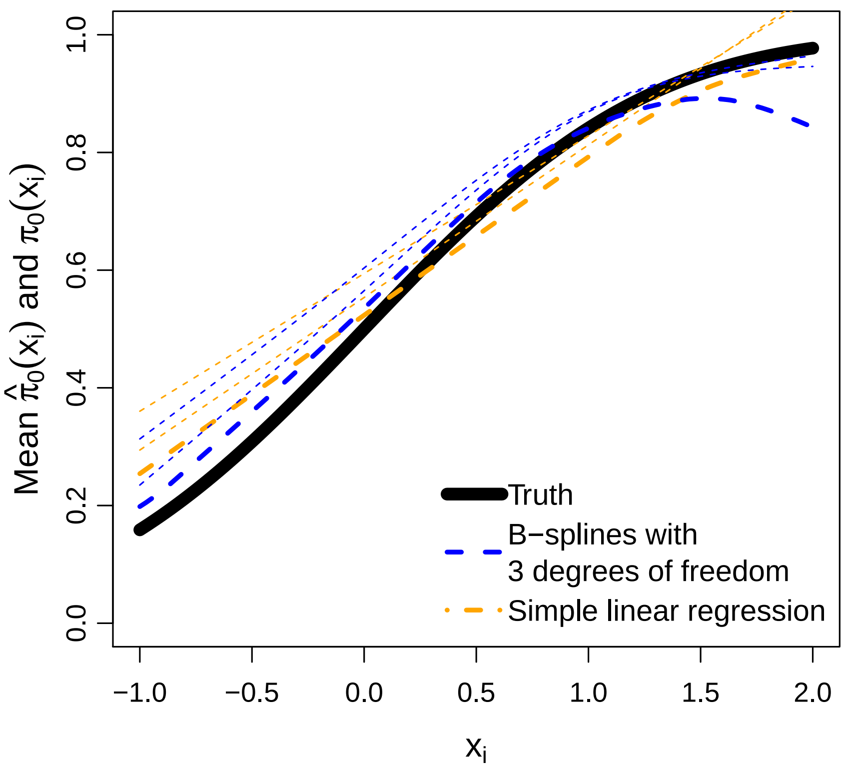
```
set.seed(1345)

## Set up the time vector and the probability of being null
tme <- seq(-1,2,length=ntest)
pi0 <- pnorm(tme)

splineMat <- ns(tme,df=3)

##save the value of pi0hat for each simulation for lambda=0.8, 0.9, final smoothed value fo
pi0hatMatLin0.8 <- pi0hatMatLin0.9 <- pi0hatMatLinFinal <-
  pi0hatMatSpl0.8 <- pi0hatMatSpl0.9 <- pi0hatMatSplFinal <-
  matrix(NA, nrow=nSims, ncol=ntest)

for(sim in 1:nSims)
  {
  ##generate p-values
  pValues <- genPvals(pi0, shape2)

  ## Get the estimates
  pi0Est.1 <- estPi0Reg(pValues, X=tme, smooth.df=3)
  pi0Est.2 <- estPi0Reg(pValues, X=splineMat, smooth.df=3)

  pi0hatMatLin0.8[sim, ] <- pi0Est.1$pi0.lambda[,round(pi0Est.1$lambda,2)==0.8]
  pi0hatMatLin0.9[sim, ] <- pi0Est.1$pi0.lambda[,round(pi0Est.1$lambda,2)==0.9]
```

```
  pi0hatMatLinFinal[sim, ] <- pi0Est.1$pi0

  pi0hatMatSpl0.8[sim, ] <- pi0Est.2$pi0.lambda[,round(pi0Est.2$lambda,2)==0.8]
  pi0hatMatSpl0.9[sim, ] <- pi0Est.2$pi0.lambda[,round(pi0Est.2$lambda,2)==0.9]
  pi0hatMatSplFinal[sim, ] <- pi0Est.2$pi0
}

## Get the mean values:
pi0hatMeanLin0.8 <- colMeans(pi0hatMatLin0.8)
pi0hatMeanLin0.9 <- colMeans(pi0hatMatLin0.9)
pi0hatMeanLinFinal <- colMeans(pi0hatMatLinFinal)

pi0hatMeanSpl0.8 <- colMeans(pi0hatMatSpl0.8)
pi0hatMeanSpl0.9 <- colMeans(pi0hatMatSpl0.9)
pi0hatMeanSplFinal <- colMeans(pi0hatMatSplFinal)

##Get the variances:
pi0hatVarLin0.8 <- apply(pi0hatMatLin0.8, 2, var)
pi0hatVarLin0.9 <- apply(pi0hatMatLin0.9, 2, var)
pi0hatVarLinFinal <- apply(pi0hatMatLinFinal, 2, var)

pi0hatVarSpl0.8 <- apply(pi0hatMatSpl0.8, 2, var)
pi0hatVarSpl0.9 <- apply(pi0hatMatSpl0.9, 2, var)
pi0hatVarSplFinal <- apply(pi0hatMatSplFinal, 2, var)

##Get the variance bounds:
# zMat <- splineMatInt
# S <- zMat%*%solve(t(zMat)%*%zMat)%*%t(zMat)
# pi0hatVarBoundFitSpl <-
#   diag(S)/(4*(1-lambda)^2)
#
# zMat <- cbind(1, tme)
# S <- zMat%*%solve(t(zMat)%*%zMat)%*%t(zMat)
# pi0hatVarBoundFitLin <-
#   diag(S)/(4*(1-lambda)^2)
```

2.1 Plot for means

```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0 ~ tme,col="black",type="l",lwd=8, lty=1,
     xlab="", yaxt = "n",
     ylim=c(0,1), ylab="")
```

```r
mtext(expression(x[i]), 1, line=3, cex=1.3)
mtext(expression(paste("Mean ", hat(pi)[0](x[i])," and ", pi[0](x[i]))), 2, line=2, cex=1.3)
points(pi0hatMeanLin0.8 ~ tme,col="orange",type="l",lwd=1, lty=2)
points(pi0hatMeanLin0.9 ~ tme,col="orange",type="l",lwd=1, lty=2)
points(pi0hatMeanLinFinal ~ tme,col="orange",type="l",lwd=3, lty=2)
points(pi0hatMeanSpl0.8 ~ tme,col="blue",type="l",lwd=1, lty=2)
points(pi0hatMeanSpl0.9 ~ tme,col="blue",type="l",lwd=1, lty=2)
points(pi0hatMeanSplFinal ~ tme,col="blue",type="l",lwd=3, lty=2)

legend("bottomright", ##x=-100, y=0.3,
       legend=c("Truth", "B-splines with\n3 degrees of freedom", "Simple linear regression")
       col=c("black", "blue", "orange"), bty="n",
       lwd=c(8,3,3), lty=c(1,2,4),
       cex=1.2, x.intersp=0.2, y.intersp=1.0)
axis(side=2, at=(0:5)/5, mgp=c(3, 0.7, 0))
```



2.2 Plots for variances

```
# par(cex.axis = 1.1, cex.main=1.3)
#
# plot(pi0hatVarBoundFitLin ~ tme, col="red", ylim=c(0, max(pi0hatVarBoundFitLin)),
#      lwd=3, lty=3,
#      type="l",
#      xlab="", ylab="")
# mtext(expression(x[i]), 1, line=3, cex=1.3)
# mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]))
# points(pi0hatVarFitLin ~ tme, col="black", type="l", lwd=3, lty=1)
# legend("top", ##x=-0.4, y=0.2,
#        legend=c("Empirical variance", "Upper bound"),
#        col=c("black", "red"), bty="n",
#        lwd=c(3,3), lty=c(1,3),
#        cex=1.2, x.intersp=0.2, y.intersp=1.0)
```

```
# par(cex.axis = 1.1, cex.main=1.3)
#
# plot(pi0hatVarBoundFitSpl ~ tme, col="red", ylim=c(0, max(pi0hatVarBoundFitSpl)),
#      lwd=3, lty=3,
#      type="l",
#      xlab="", ylab="")
# mtext(expression(x[i]), 1, line=3, cex=1.3)
# mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i]))
# points(pi0hatVarFitSpl ~ tme, col="black", type="l", lwd=3, lty=1)
# legend("top", ##x=-0.4, y=0.2,
#        legend=c("Empirical variance", "Upper bound"),
#        col=c("black", "red"), bty="n",
#        lwd=c(3,3), lty=c(1,3),
#        cex=1.2, x.intersp=0.2, y.intersp=1.0)
```

3 Probability of being a false positive as a sine + step function

```
set.seed(1345)

## Set up the time vector and the probability of being null
tme1 <- seq(-1*pi,2*pi,length=ntest)
tme2 <- rep(1:0, each=ntest/2)
pi0 <- 1/4*sin(tme1) + tme2/4 + 1/2
```

```r
##pi0 <- 1/4*sin(tme1) + 0.5
range(pi0)
```

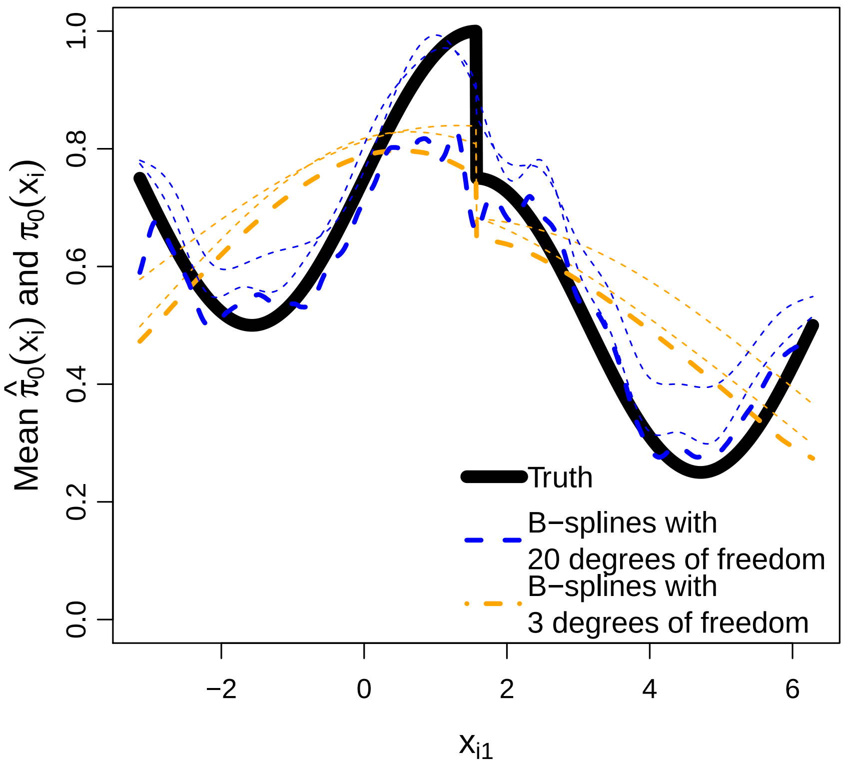
```
## [1] 0.2500028 0.9999972
```

```r
splineMat3 <- cbind(ns(tme1,df=3), tme2)
splineMat20 <- cbind(ns(tme1,df=20), tme2)

##save the value of pi0hat for each simulation for lambda=0.8, 0.9, final smoothed value fo
pi0hatMat3.0.8 <- pi0hatMat3.0.9 <- pi0hatMat3.Final <-
  pi0hatMat20.0.8 <- pi0hatMat20.0.9 <- pi0hatMat20.Final <-
  matrix(NA, nrow=nSims, ncol=ntest)

for(sim in 1:nSims)
{
  ##generate p-values
  pValues <- genPvals(pi0, shape2)

  ## Get the estimates
  pi0Est.1 <- estPi0Reg(pValues, X=splineMat3, smooth.df=3)
  pi0Est.2 <- estPi0Reg(pValues, X=splineMat20, smooth.df=3)

  pi0hatMat3.0.8[sim, ] <- pi0Est.1$pi0.lambda[,round(pi0Est.1$lambda,2)==0.8]
  pi0hatMat3.0.9[sim, ] <- pi0Est.1$pi0.lambda[,round(pi0Est.1$lambda,2)==0.9]
  pi0hatMat3.Final[sim, ] <- pi0Est.1$pi0

  pi0hatMat20.0.8[sim, ] <- pi0Est.2$pi0.lambda[,round(pi0Est.2$lambda,2)==0.8]
  pi0hatMat20.0.9[sim, ] <- pi0Est.2$pi0.lambda[,round(pi0Est.2$lambda,2)==0.9]
  pi0hatMat20.Final[sim, ] <- pi0Est.2$pi0
 }

## Get the mean values:
pi0hatMean3.0.8 <- colMeans(pi0hatMat3.0.8)
pi0hatMean3.0.9 <- colMeans(pi0hatMat3.0.9)
pi0hatMean3.Final <- colMeans(pi0hatMat3.Final)

pi0hatMean20.0.8 <- colMeans(pi0hatMat20.0.8)
pi0hatMean20.0.9 <- colMeans(pi0hatMat20.0.9)
pi0hatMean20.Final <- colMeans(pi0hatMat20.Final)

##Get the variances:
pi0hatVar3.0.8 <- apply(pi0hatMat3.0.8, 2, var)
pi0hatVar3.0.9 <- apply(pi0hatMat3.0.9, 2, var)
pi0hatVar3.Final <- apply(pi0hatMat3.Final, 2, var)

pi0hatVar20.0.8 <- apply(pi0hatMat20.0.8, 2, var)
```

```
pi0hatVar20.0.9 <- apply(pi0hatMat20.0.9, 2, var)
pi0hatVar20.Final <- apply(pi0hatMat20.Final, 2, var)

# ##Get the variance bounds:
# zMat <- splineMatInt3
# S <- zMat%*%ginv(t(zMat)%*%zMat)%*%t(zMat)
# pi0hatVarBound3 <-
#   diag(S)/(4*(1-lambda)^2)
#
# zMat <- splineMatInt20
# S <- zMat%*%ginv(t(zMat)%*%zMat)%*%t(zMat)
# pi0hatVarBound20 <-
#   diag(S)/(4*(1-lambda)^2)
```

3.1 Plot for means

```
par(cex.axis = 1.1, cex.main=1.3)

plot(pi0 ~ tme1,col="black",type="l",lwd=8, lty=1,
     xlab="", yaxt = "n",
     ylim=c(0,1), ylab="")
mtext(expression(x[i1]), 1, line=3, cex=1.3)
mtext(expression(paste("Mean ", hat(pi)[0](x[i]),"  and ",
                       pi[0](x[i]))), 2, line=2, cex=1.3)
points(pi0hatMean3.0.8 ~ tme1,col="orange",type="l",lwd=1, lty=2)
points(pi0hatMean3.0.9 ~ tme1,col="orange",type="l",lwd=1, lty=2)
points(pi0hatMean3.Final ~ tme1,col="orange",type="l",lwd=3, lty=2)
points(pi0hatMean20.0.8 ~ tme1,col="blue",type="l",lwd=1, lty=2)
points(pi0hatMean20.0.9 ~ tme1,col="blue",type="l",lwd=1, lty=2)
points(pi0hatMean20.Final ~ tme1,col="blue",type="l",lwd=3, lty=2)

legend("bottomright", ##x=-100, y=0.3,
       legend=c("Truth", "B-splines with\n20 degrees of freedom",
                "B-splines with\n3 degrees of freedom"),
       col=c("black", "blue", "orange"), bty="n",
       lwd=c(8,3,3), lty=c(1,2,4),
       cex=1.2, x.intersp=0.2, y.intersp=1.15)
axis(side=2, at=(0:5)/5, mgp=c(3, 0.7, 0))
```



3.2 Plots for variances

```
# par(cex.axis = 1.1, cex.main=1.3)
#
# plot(pi0hatVarBound3 ~ tme1, col="red", ylim=c(0, max(pi0hatVarBound3)),
#      lwd=3, lty=3,
#      type="l",
#      xlab="", ylab="")
# mtext(expression(x[i1]), 1, line=3, cex=1.3)
# mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i])
# points(pi0hatVar3 ~ tme1, col="black", type="l", lwd=3, lty=1)
# legend("top", ##x=-0.4, y=0.2,
#        legend=c("Empirical variance", "Upper bound"),
#        col=c("black", "red"), bty="n",
#        lwd=c(3,3), lty=c(1,3),
```

```
#        cex=1.2, x.intersp=0.2, y.intersp=1.0)
```

```
# par(cex.axis = 1.1, cex.main=1.3)
#
# plot(pi0hatVarBound20 ~ tme1, col="red", ylim=c(0, max(pi0hatVarBound20)),
#      lwd=3, lty=3,
#      type="l",
#      xlab="", ylab="")
# mtext(expression(x[i1]), 1, line=3, cex=1.3)
# mtext(expression(paste("Variance and upper bound of variance for ", " ", hat(pi)[0](x[i])
# points(pi0hatVar20 ~ tme1, col="black", type="l", lwd=3, lty=1)
# legend("top", ##x=-0.4, y=0.2,
#        legend=c("Empirical variance", "Upper bound"),
#        col=c("black", "red"), bty="n",
#        lwd=c(3,3), lty=c(1,3),
#        cex=1.2, x.intersp=0.2, y.intersp=1.0)
```

Session info:

```
devtools::session_info()

## Session info ----------------------------------------------

##  setting  value
##  version  R version 3.3.1 (2016-06-21)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.1252
##  tz       America/New_York
##  date     2016-07-28

## Packages --------------------------------------------------

##  package  * version date       source
##  devtools   1.12.0  2016-06-24 CRAN (R 3.3.1)
##  digest     0.6.9   2016-01-08 CRAN (R 3.3.1)
##  evaluate   0.9     2016-04-29 CRAN (R 3.3.1)
##  formatR    1.4     2016-05-09 CRAN (R 3.3.1)
##  highr      0.6     2016-05-09 CRAN (R 3.3.1)
##  knitr    * 1.13    2016-05-09 CRAN (R 3.3.1)
##  magrittr   1.5     2014-11-22 CRAN (R 3.3.1)
##  MASS     * 7.3-45  2016-04-21 CRAN (R 3.3.1)
##  memoise    1.0.0   2016-01-29 CRAN (R 3.3.1)
##  stringi    1.1.1   2016-05-27 CRAN (R 3.3.0)
##  stringr    1.0.0   2015-04-30 CRAN (R 3.3.1)
##  withr      1.0.2   2016-06-20 CRAN (R 3.3.1)
```