Hands-on with COMBSS

Step-by-step R quick-start for the StatFest 2026 talk

Sarat Moka

2026-05-22

Table of contents

1 One-time setup

These steps assume a Mac or Linux laptop with conda installed.

1.1 Create an isolated conda environment

```
conda create -n combss python=3.12 -y
conda activate combss
```

The env is named `combss`. Python 3.12 is used here only because conda needs *some* Python; the rest of this handout is R.

1.2 Install R inside the env

```
conda install -c conda-forge r-base r-essentials -y
```

1.3 Install RStudio Desktop

RStudio Desktop is **not** installed via conda — download the installer: https://posit.co/download/rstudio-desktop and drag to Applications.

RStudio will auto-detect the conda R interpreter once the env is active. If it doesn't, set the path manually under **Tools → Global Options → General → R version**, using the path returned by `which R` after `conda activate combss`.

1.4 Install the two R packages used in this handout

In the RStudio console (with the conda R active):

```
install.packages(c("combss", "glmnet"))
```

If you are prompted:

> *There is a binary version available but the source version is later. Do you want to install from sources the package which needs compilation?*

Answer **no** — the binary is fine and avoids a Fortran-toolchain dependency.

Sanity check:

```
library(combss)
packageVersion("combss")     # expect >= 0.1.0
```

1.5 Get the Khan SRBCT data (needed for Demo 3)

Run these in a **terminal** (not the RStudio console):

```
git clone https://github.com/saratmoka/combss-statfest.git
cd combss-statfest
```

Then in RStudio: **Session → Set Working Directory → Choose Directory...**, pointing at the cloned `combss-statfest` folder.

2 Demo 1 — Linear simulation

Goal. Five truly active predictors out of thirty. COMBSS should return $\{1, 2, 3, 4, 5\}$ at $k = 5$.

```
library(combss)

set.seed(2026)
n <- 300; p <- 30
beta <- c(3, 2, 1.5, 1, 0.5, rep(0, p - 5))
x <- matrix(rnorm(n * p), n, p)
y <- as.numeric(x %*% beta + rnorm(n) * 0.5)
itr <- 1:200; iva <- 201:300

fit <- combss(x[itr, ], y[itr],
              x_val = x[iva, ], y_val = y[iva],
              family = "gaussian", q = 10)

fit$best_k                    # expect 5
fit$subset_list[1:6]          # nested supports building to {1,2,3,4,5}

plot(seq_along(fit$mse_path), fit$mse_path, type = "b",
     pch = 19, xlab = "k", ylab = "Validation MSE")
abline(v = fit$best_k, lty = 2)
```

3 Demo 2 — High-dimensional logistic ($p \gg n$)

Goal. $n = 200$, $p = 1000$, ten truly active predictors with AR(1) correlation. COMBSS should recover the true support at $k = 10$.

```
library(combss)

set.seed(2026)
n <- 200; p <- 1000; k0 <- 10
rho <- 0.5
ar1_sqrt <- function(p, rho) {
  Sigma <- rho ^ abs(outer(1:p, 1:p, "-"))
  ev <- eigen(Sigma, symmetric = TRUE)
  ev$vectors %*% diag(sqrt(pmax(ev$values, 0))) %*% t(ev$vectors)
}
L <- ar1_sqrt(p, rho)
z <- matrix(rnorm(n * p), n, p)
x <- z %*% L
beta <- c(rep(1, k0), rep(0, p - k0))
eta <- as.numeric(x %*% beta)
y <- rbinom(n, 1, plogis(eta))

itr <- 1:150; iva <- 151:200
fit_hd <- combss(x[itr, ], y[itr],
                 x_val = x[iva, ], y_val = y[iva],
                 family = "binomial", q = 15)

fit_hd$subset_list[[10]]     # the model at k = 10 - should contain 1..10

tp_path <- sapply(fit_hd$subset_list,
                  function(s) length(intersect(s, 1:k0)))
plot(seq_along(tp_path), tp_path, type = "b", pch = 19,
     xlab = "k", ylab = "True positives (out of 10)",
     ylim = c(0, k0 + 1))
abline(h = k0, lty = 2, col = "grey")
```

4 Demo 3 — Khan SRBCT (multinomial, $p = 2308$, 4 classes)

Goal. Pick a small panel of genes that perfectly classifies four cancer subtypes on a held-out test set of size 20.

Working directory must contain the `data/` folder from the repo.

```
library(combss)

train <- read.csv("data/Khan_train.csv")
test  <- read.csv("data/Khan_test.csv")
x_train <- as.matrix(train[, -1]); y_train <- train$y
x_test  <- as.matrix(test[, -1]);  y_test  <- test$y

fit_khan <- combss(x_train, y_train,
                   x_val = x_test, y_val = y_test,
                   family = "multinomial", q = 20)

fit_khan$best_k                                    # expect 12
fit_khan$subset_list[[ fit_khan$best_k ]]          # the 12 selected genes

plot(seq_along(fit_khan$accuracy_path), fit_khan$accuracy_path,
     type = "b", pch = 19, xlab = "k", ylab = "Test accuracy")
abline(v = fit_khan$best_k, lty = 2)
```

5 Demo 4 — Head-to-head with the lasso

Goal. Run COMBSS and `cv.glmnet` on the **same** simulated data (identical to Demo 1) and read off the difference.

```
library(combss)
library(glmnet)

# Identical data to Demo 1
set.seed(2026)
n <- 300; p <- 30
beta <- c(3, 2, 1.5, 1, 0.5, rep(0, p - 5))
x <- matrix(rnorm(n * p), n, p)
y <- as.numeric(x %*% beta + rnorm(n) * 0.5)
itr <- 1:200; iva <- 201:300

# Lasso
cv_l <- cv.glmnet(x[itr, ], y[itr], alpha = 1)
beta_l <- as.numeric(coef(cv_l, s = "lambda.min"))[-1]
sel_l  <- which(beta_l != 0)
pred_l <- predict(cv_l, newx = x[iva, ], s = "lambda.min")
mse_l  <- mean((y[iva] - pred_l)^2)
cat(sprintf("Lasso  : selected = %d, val_mse = %.4f\n",
            length(sel_l), mse_l))

# COMBSS
fit <- combss(x[itr, ], y[itr],
              x_val = x[iva, ], y_val = y[iva],
              family = "gaussian", q = 10)
sel_c <- fit$subset_list[[fit$best_k]]
mse_c <- min(fit$mse_path)
cat(sprintf("COMBSS : selected = %d, val_mse = %.4f\n",
            length(sel_c), mse_c))
```

Expected. Lasso picks ~16 features for a validation MSE of about 0.24; COMBSS picks exactly 5 for a validation MSE of about 0.21.

6 Where to go next

- **Talk companion site** — https://saratmoka.github.io/combss-statfest contains the same demos plus the methodology pages, comparisons against SCAD and MCP, and the rice GWAS application.
- **Python version of this handout** — a parallel `handout-python.pdf` walks through the same demos using `combss` from PyPI inside Spyder.
- **Source** — https://github.com/saratmoka/combss-statfest.