Reproduce paper plots with scPower

Matthias Heinig, Katharina Schmid

04 August, 2020

Contents

This vignette shows how the plots of the paper "Design and power analysis for multi-sample single cell genomics experiments" can be reproduced using scPower. For an introduction into the package with detailed explanations, please have a look at the second vignette "introduction-scPower". Due to runtime reasons, not the complete analysis is performed in this vignette and instead sometimes precalculated values provided from the package. However, for all steps it is explained how the precalculated values are generated. For more information on the modelling and all citations, please have a look into our publication.

We imported additionally the same color schemes and plotting packages (ggplot2 and ggpubr) to be able to create exactly the same plots as in the publication.

```
library(scPower)
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.5.2
library(data.table)
#> Warning: package 'data.table' was built under R version 3.5.2
library(viridis)
#> Loading required package: viridisLite
```

```
library(ggpubr)
library(RColorBrewer)
```

```
#Color specifications
col.paired <- brewer.pal(n = 12, "Paired")
col.set <- col.paired[c(2,8,9)]
col.set2 <- col.paired[c(9,5,7,3,8,2)]
col.set3 <- brewer.pal(n = 11, "PRGn")[c(4,6,8)]
col.set3 <- c(col.set[3], "#F7F7F7", col.set[1])
col.set4 <- c(col.set[1], col.set[3])
col.set5<-col.paired[seq(2,12,2)]
```

Paper Plot

Figure 1: Dependence of experimental design parameters

Figure 1 is a schematic overview figure showing the dependence of experimental design parameters.

Figure 2: Expression probability model parameterized by UMI counts per cell

Figure 2 shows how the fit of our expression probabilty model is applied on two example data sets, our own data set and one of Kang et al, 2018. The model is fitted using our own PBMC data set. The fits are provided in the scPower package in the data frames "disp.fun.param" and "gamma.mixed.fits". Also the used UMI-read fit was generated using our own PBMC data set, saved in the data frame "read.umi.fit" as "10X_PBMC_1". The fitting itself is not shown here due to runtime limitations, but the fits were exactly produced as described in the introduction-scPower" vignette in section "Expression probability curves" (using the same functions in the same order). The only difference is that in the introduction vignette it is only shown for one cell type, we performed the fit separately for all celltypes with at least 50 cells.

The observed expression curves were calculated as described in section "Counting observed expressed genes", the percalculated values are saved due to time reasons in the data frame "observed.gene.counts".

```
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

#Selected run
run<-"Run 5"

###############
#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count10" &
    observed.gene.counts$run==run,]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){
  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=14,
    nCellsCt=estimates.allSamples$num.cells[i]/14,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
```

```
    disp.fun.param = scPower::disp.fun.param,
    min.counts=10,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))
#Set the ordering of expressed vs estimated correctly
plot.estimates$variable<-factor(plot.estimates$variable,
                                levels=c("expressed.genes","estimated.counts"))

#Replace subsampling
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75","subsampled50",
                             "subsampled25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]

g.run.10<-ggplot(data=plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8))+
  scale_fill_manual(values=col.set2)+
  scale_color_manual("Read depth",values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.position="none",
        aspect.ratio = 0.8,
        legend.direction = "vertical", legend.box = "horizontal")


###############
#Fit for own data and count > 0
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count0" &
    observed.gene.counts$run==run,]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){
  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=14,
    nCellsCt=estimates.allSamples$num.cells[i]/14,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
```

```
    min.counts=0,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Set the ordering of expressed vs estimated correctly
plot.estimates$variable<-factor(plot.estimates$variable,
                                levels=c("expressed.genes","estimated.counts"))

#Replace subsampling
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75","subsampled50",
                             "subsampled25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]

g.run.0<-ggplot(data=plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8))+
  scale_fill_manual(values=col.set2)+
  scale_color_manual("Read depth",values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.position="none",
        aspect.ratio = 0.8,
        legend.direction = "vertical", legend.box = "horizontal")

###############
#Fit for Kang data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="Kang_count10",]

personsPerRunKang<-list("A"=4,"B"=4,"C"=8)

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){
  sampleSize<-personsPerRunKang[[estimates.allSamples$run[i]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
```

```
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=10,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Remove dendritic cell type (only fitted with data from the overloaded run)
plot.estimates<-plot.estimates[plot.estimates$cell.type!="Dendritic cells",]

g.ye.10<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=run,linetype=variable))+
  geom_point(aes(shape=cell.type,color=run,fill=run),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=col.set2)+
  scale_color_manual("Batch",values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text.x=element_text(size=8),
        legend.position="none",
        aspect.ratio = 0.8)

###############
#Fit for Kang data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="Kang_count0",]

personsPerRunKang<-list("A"=4,"B"=4,"C"=8)

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){
  sampleSize<-personsPerRunKang[[estimates.allSamples$run[i]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=0,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))
```

```
}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Remove dendritic cell type (only fitted with data from the overloaded run)
plot.estimates<-plot.estimates[plot.estimates$cell.type!="Dendritic cells",]

g.ye.0<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=run,linetype=variable))+
  geom_point(aes(shape=cell.type,color=run,fill=run),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=col.set2)+
  scale_color_manual("Batch",values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text.x=element_text(size=8),
        legend.position="none",
        aspect.ratio = 0.8)

#Create a plot only for the legend and save that
g.legend<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=run,linetype=variable))+
  geom_point(aes(shape=cell.type,color=run,fill=run),size=2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  theme_bw()+
theme(legend.position = "bottom",
      legend.title=element_text(size=8),
      legend.text=element_text(size=8))+
  guides(linetype=FALSE,color=FALSE,fill=FALSE)

leg.ct<-get_legend(g.legend)
leg.ct<-as_ggplot(leg.ct)

g.plots<-ggarrange(g.run.10,g.run.0,g.ye.10,g.ye.0, ncol=2, nrow=2,
          labels=c("A","B","C","D"))

g<-ggarrange(g.plots,leg.ct,nrow=2,heights=c(4,0.3))
print(g)
```

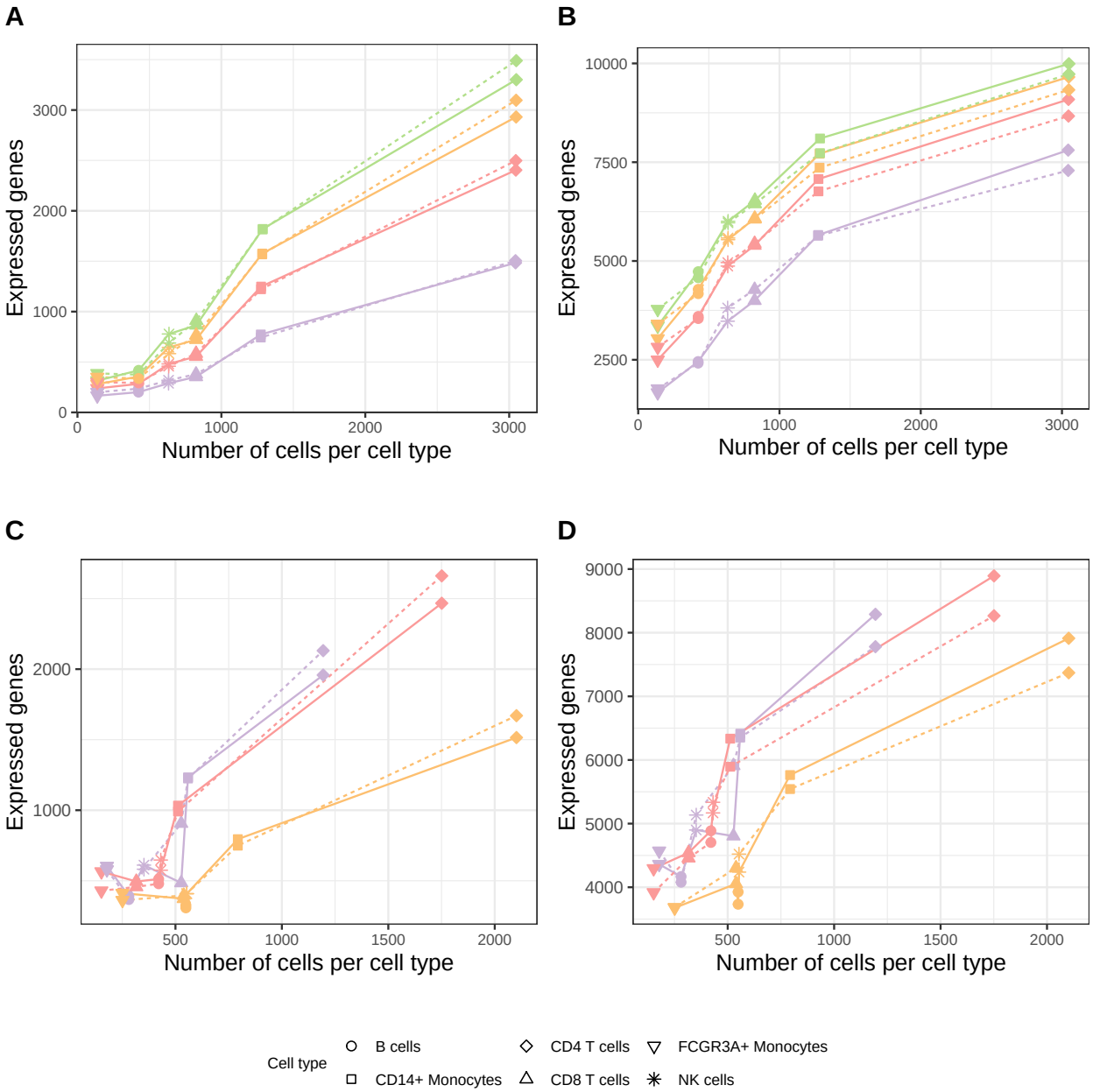

Figure 2: Expression probability model parameterized by UMI counts per cell. Plot A and B show the observed (solid) number of expressed genes and the number of expressed genes expected under our model (dashed) on the y-axis and the number of cells per cell type (cell type indicated by the point symbol) on the x-axis for Run 5 of the PBMC data set (Additional file 1: Table S1). The data is subsampled to different read depths (indicated by the color). Similarly, plot C and D show expressed genes for the three batches of the Ye data set. The definition for an expressed gene is based on a flexible user defined cutoff. Here it was parameterized: in Figure A and C, a gene is called expressed with count > 10 in more than 50% of the individuals, in Figure B and D, count > 0 in more than 50% of the individuals.

Figure 3: Expression probability, DE/eQTL power and overall detection power

The figure shows the influence of the experimental parameters on the detection power. The same expression probability model as in Figure 2 is used, the priors for the power, i.e. the effect sizes and expression ranks, are taken from different reference studies (see publication for citation and more explanation). The experimental

parameters are set to realistic values. FDR adjustment is set as multiple testing correction for DE and FWER adjustment for eQTLs.

```
#General parameters
transcriptome.mapped.reads<-20000
nGenes<-21000
ct<-"CD4 T cells"
ct.freq<-1
cellsPerLane<-20000

########
# DE subplot
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(4,6,8,10,16,20)
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                                   param.combis$cellsPerCelltype[i],
                                   transcriptome.mapped.reads,
                                   ct.freq, "de",de.ref.study,ref.study.name,
                                   cellsPerLane,read.umi.fit[
                                     read.umi.fit$type=="10X_PBMC_1",],
                                   gamma.mixed.fits,ct,
                                   disp.fun.param,mappingEfficiency = 1,
                                   multipletRate=0,multipletFactor=1,
                                   min.UMI.counts = 10,
                                   perc.indiv.expr = 0.5,
                                   nGenes=21000,
                                   sign.threshold = 0.05,
                                   MTmethod = "FDR"))
#> Loading required package: MKmisc
#> Warning: package 'MKmisc' was built under R version 3.5.2

power.DE.study<-rbindlist(res)

#Plot DE results
power.DE.study<-reshape2::melt(power.DE.study,
                               id.vars=c("name","sampleSize","totalCells",
                                         "usableCells","multipletFraction",
                                         "ctCells","readDepth",
                                         "readDepthSinglet","mappedReadDepth",
                                         "expressedGenes"))
power.DE.study$variable<-factor(power.DE.study$variable,
                                levels=c("exp.probs","power","powerDetect"))

var.labs<-c("Expression probability", "DE power",
            "Detection power")
names(var.labs)<-c("exp.probs","power","powerDetect")
```

```
power.DE.study$ctCells<-as.factor(power.DE.study$ctCells)
power.DE.study$sampleSize<-as.factor(power.DE.study$sampleSize)
g.DE<-ggplot(power.DE.study,aes(x=ctCells,y=sampleSize,fill=value))+
  geom_tile()+
  facet_wrap(~variable,ncol=3,labeller = labeller(variable=var.labs))+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Probability",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

########
# eQTL subplot
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(50,100,150,200)
ref.study.name<-"Blueprint (Tcells)"

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                                   param.combis$cellsPerCelltype[i],
                                   transcriptome.mapped.reads,
                                   ct.freq, "eqtl",eqtl.ref.study,ref.study.name,
                                   cellsPerLane,read.umi.fit[
                                     read.umi.fit$type=="10X_PBMC_1",],
                                   gamma.mixed.fits,ct,
                                   disp.fun.param,mappingEfficiency = 1,
                                   multipletRate=0,multipletFactor=1,
                                   min.UMI.counts = 10,
                                   perc.indiv.expr = 0.5,
                                   nGenes=21000,
                                   sign.threshold = 0.05,
                                   MTmethod = "Bonferroni"))
#> Loading required package: pwr

power.eqtl.study<-rbindlist(res)

#Plot eQTL results
power.eqtl.study<-reshape2::melt(power.eqtl.study,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth",
                                           "readDepthSinglet","mappedReadDepth",
                                           "expressedGenes"))
power.eqtl.study$variable<-factor(power.eqtl.study$variable,
                                  levels=c("exp.probs","power","powerDetect"))
```

```
var.labs<-c("Expression probability", "eQTL power",
            "Detection power")
names(var.labs)<-c("exp.probs","power","powerDetect")

power.eqtl.study$ctCells<-as.factor(power.eqtl.study$ctCells)
power.eqtl.study$sampleSize<-as.factor(power.eqtl.study$sampleSize)
g.eqtl<-ggplot(power.eqtl.study,aes(x=ctCells,y=sampleSize,fill=value))+
  geom_tile()+
  facet_wrap(~variable,ncol=3,labeller = labeller(variable=var.labs))+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Probability",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

g<-ggarrange(g.DE,g.eqtl,ncol=1,nrow=2,labels=c("A","B"),
          align="hv",common.legend=TRUE,legend="right")

print(g)
```

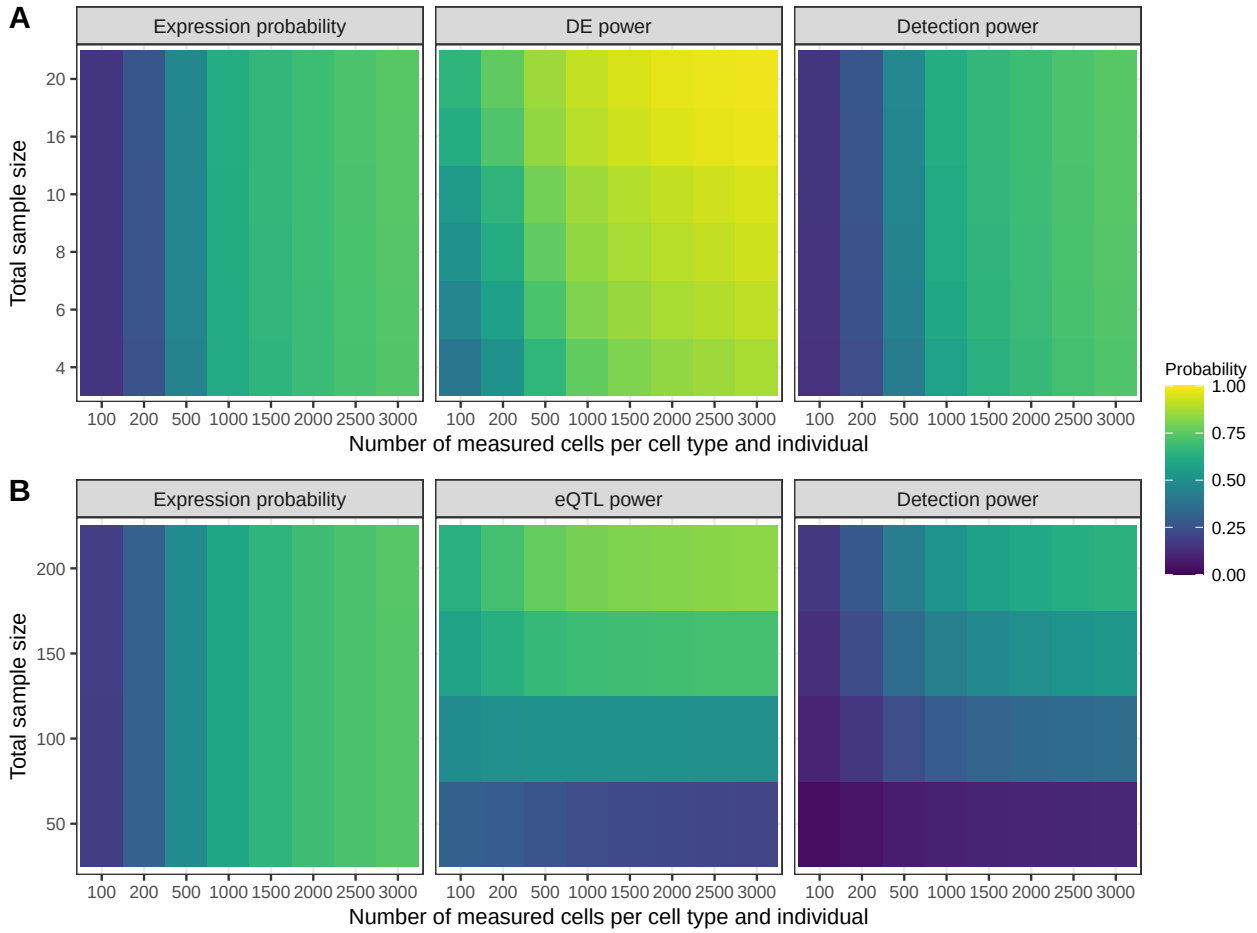

Figure 3: Expression probability, DE/eQTL power and overall detection power. Power estimation using data driven priors for A. DE genes and B. eQTL genes dependent on the total sample size and the number of measured cells per cell type. The detection power is the product of the expression probability and the power to detect the genes as DE or eQTL genes, respectively. The fold change for DEGs and the R2 for eQTL genes were taken from published studies, together with the expression rank of the genes. For A, the Blueprint CLL study with comparison iCLL vs mCLL was used, for B, the Blueprint T cell study. The expression profile and expression probabilities in a single cell experiment with a specific number of samples and measured cells was estimated using our gamma mixed models, setting the definition for expressed to > 10 counts in more than 50% of the individuals.

Figure 4: Optimal parameters for varying budgets and 10X data

How the optimal parameters develop for an increasing experimental budget, is shown in Figures 4 and 5. As this means running the optimization function many times, this takes too long to run it in the vignette. Instead, the code is given, but the evaluation set to FALSE. Interested users can run the calculation, by setting it to TRUE. Calculations are done for DE and eQTL studies, with observed and simulated priors.

```
#####
#General parameters

ct<-"CD14+ Monocytes"
ct.freq<-0.2

cellsPerChanel<-20000
costKit<-5600
costFlowCell<-14032
readsPerFlowcell<-4100*10^6

mapping.efficiency<-0.8
minUMI<-3
perc.indiv<-0.5

budget<-seq(5000,100000,by=5000)

########
#Parameters for DE studies
readDepth<-c(2000,5000,10000,15000,20000,30000,40000,50000,60000,70000,100000)
cellsPerPerson<-c(100,200,500,700,seq(1000,9000,500))
```

```
#####
#Simulated DE studies - PBMC (10X)

simulated.parameters<-data.frame(mean.value=c(2,0.5,2,0.5),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

#Number simulated DE genes
numGenes<-250

de.raw.values<-NULL
```

```
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.DE.simulation(mean=simulated.parameters$mean.value[s],
                                        sd=1,numGenes=numGenes)

  sample.DE.genes<-data.frame(ranks=ranks, FoldChange=effectSizes,
                              name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"de",ct,ct.freq,
                              costKit,costFlowCell,readsPerFlowcell,
                              sample.DE.genes,"Simulated",
                              cellsPerChanel,read.umi.fit[
                                read.umi.fit$type=="10X_PBMC_1",],
                              gamma.mixed.fits,
                              disp.fun.param,
                              nCellsRange = cellsPerPerson,
                              readDepthRange = readDepth,
                              mappingEfficiency = mapping.efficiency,
                              min.UMI.counts = minUMI,
                              perc.indiv.expr = perc.indiv,
                              MTmethod = "FDR",
                              speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  de.raw.values<-rbind(de.raw.values,max.values)
}

######
# DE studies with observed priors - PBMC (10X)

#Filter the DE reference study for PBMC data sets
de.ref.study<-scPower::de.ref.study[de.ref.study$type=="PBMC",]

de.real.values<-NULL
for(studyName in unique(de.ref.study$name)){

  max.values<-NULL
```

```
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"de",ct,ct.freq,
                  costKit,costFlowCell,readsPerFlowcell,
                  de.ref.study,studyName,
                  cellsPerChanel,read.umi.fit[read.umi.fit$type=="10X_PBMC_1",],
                  gamma.mixed.fits,
                  disp.fun.param,
                  nCellsRange = cellsPerPerson,
                  readDepthRange = readDepth,
                  mappingEfficiency = mapping.efficiency,
                  min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                  MTmethod = "FDR",
                  speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  de.real.values<-rbind(de.real.values,max.values)
}
```

```
#Parameters for eQTL studies
readDepth<-c(2000,5000,10000,15000,20000,30000,40000,50000,60000,70000,100000)
cellsPerPerson<-seq(500,9000,500)
```

```
#####
#Simulated eQTL studies - PBMC (10X)

simulated.parameters<-data.frame(mean.value=c(0.5,0.2,0.5,0.2),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

#Number simulated eQTL genes
numGenes<-2000

eqtl.raw.values<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.eQTL.simulation(mean=simulated.parameters$mean.value[s],
                                          sd=0.2,numGenes=numGenes)
```

```
  sample.eqtl.genes<-data.frame(ranks=ranks,Rsq=effectSizes,name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"eqtl",ct,ct.freq,
                          costKit,costFlowCell,readsPerFlowcell,
                          sample.eqtl.genes,"Simulated",
                          cellsPerChanel,
                          read.umi.fit[read.umi.fit$type=="10X_PBMC_1",],
                          gamma.mixed.fits,
                          disp.fun.param,
                          nCellsRange = cellsPerPerson,
                          readDepthRange = readDepth,
                          mappingEfficiency = mapping.efficiency,
                          min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                          MTmethod = "Bonferroni",
                          speedPowerCalc=TRUE))

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  eqtl.raw.values<-rbind(eqtl.raw.values,max.values)
}



######
# eQTL studies with observed priors - PBMC (10X)

eqtl.real.values<-NULL
for(studyName in unique(eqtl.ref.study$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.restrictedDoublets(i,"eqtl",ct,ct.freq,
                  costKit,costFlowCell,readsPerFlowcell,
                  eqtl.ref.study,studyName,
                  cellsPerChanel,read.umi.fit[read.umi.fit$type=="10X_PBMC_1",],
                  gamma.mixed.fits,
                  disp.fun.param,
                  nCellsRange = cellsPerPerson,
                  readDepthRange = readDepth,
                  mappingEfficiency = mapping.efficiency,
                  min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                  MTmethod = "Bonferroni",
                  speedPowerCalc=TRUE))

    max.values<-rbind(max.values,
```

```
                        power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  eqtl.real.values<-rbind(eqtl.real.values,max.values)
}
```

As mentioned above, percalculated values are loaded in the vignette instead, created with the code shown above (but set to eval=FALSE).

```
#Get precalculated values
data(precalculatedBudgetOptim) #budget.optimization
eqtl.raw.values<-budget.optimization[budget.optimization$type=="eQTL_simulated_PBMC",]
de.raw.values<-budget.optimization[budget.optimization$type=="DE_simulated_PBMC",]
eqtl.real.values<-budget.optimization[budget.optimization$type=="eQTL_observed_PBMC",]
de.real.values<-budget.optimization[budget.optimization$type=="DE_observed_PBMC",]

#Replace probability names
var.labs<-setNames(c("Detection power","Cells per individual",
                     "Read depth","Sample size"),
                   c("powerDetect","totalCells","readDepth","sampleSize"))

#Data frame with dummy values to scale the y axes
dummy.plot<-data.frame(budget=c(min(budget),max(budget),min(budget),max(budget),
                                min(budget),max(budget),min(budget),max(budget)),
                       variable=c("powerDetect","powerDetect","readDepth",
                                  "readDepth","totalCells","totalCells",
                                  "sampleSize","sampleSize"),
                       value=c(0,1,min(readDepth),max(readDepth),
                               min(cellsPerPerson),max(cellsPerPerson),0,400))

#eQTL simulated values
plot.values<-eqtl.raw.values[,c("budget","powerDetect","totalCells","readDepth",
                                "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.eqtl.1<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#DE simulated values
plot.values<-de.raw.values[,c("budget","powerDetect","totalCells","readDepth",
                              "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.de.1<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
```

```
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#eQTL real values
plot.values<-eqtl.real.values[,c("budget","powerDetect","totalCells","readDepth",
                                 "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.eqtl.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#DE real values
plot.values<-de.real.values[,c("budget","powerDetect","totalCells","readDepth",
                               "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.de.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#Overview plot
g<-ggarrange(g.de.1,g.eqtl.1,g.de.2,g.eqtl.2,
             ncol=1,nrow=4,labels=c("A","B","C","D"),align="hv")
print(g)
```

Figure 4: Optimal parameters for varying budgets and 10X Genomics data. The figure shows the maximal reachable detection power (y-axis, first column) for a given experimental budget (x-axis) and the corresponding optimal parameter combinations for that budget (y-axis, second till fourth column). The colored lines indicate different effect sizes and gene expression rank distributions. Subplots A-B visualize different simulated effect sizes and rank distributions (simulation names see text) for DEG studies (A) and eQTL studies (B) with models fitted on 10X PBMC data. Subplots C-D visualize effect sizes and rank distributions observed in cell type sorted bulk RNA-seq DEG studies (C) and eQTL studies (D) with model fits analogously to A-B.

Figure 5: Optimal parameters for varying budgets and Drop-seq and Smart-seq data

The same evaluation is done in Figure 5 than in Figure 4, only this time with priors of Drop-seq lung data and Smart-seq pancreas data instead of 10X Genomics PBMC data. Similar to Figure 4, the code for the calculation is given, but set to eval=FALSE due to long runtime. Instead, precalculated values are taken for the plotting.

```
#Drop-seq parameters

#Cell type
ct<-"NK cell"
ct.freq<-0.2

cellsPerChanel<-NA
libPrepCell<-0.09
costFlowCell<-14032
readsPerFlowcell<-4100*10^6
```

```
readDepth<-c(2000,5000,10000,15000,20000,30000,40000,50000,60000,70000,100000)
cellsPerPerson<-c(100,200,500,700,seq(1000,9000,500))

mapping.efficiency<-0.8
nFittedGenes<-24000

#Model growth rate constant
multipletRateGrowth<-"constant"
multipletRate<-0.05

#Number of DE genes to be modeled
numGenes<-250
```

```
#####
#Simulated DE studies - lung (drop-seq)

simulated.parameters<-data.frame(mean.value=c(2,0.5,2,0.5),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

de.raw.values.lung<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.DE.simulation(mean=simulated.parameters$mean.value[s],
                                        sd=1, numGenes=numGenes)

  sample.DE.genes<-data.frame(ranks=ranks, FoldChange=effectSizes,
                              name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.libPrepCell(i,"de",ct,ct.freq,
                             libPrepCell,costFlowCell,readsPerFlowcell,
                             sample.DE.genes,"Simulated",
                             cellsPerChanel,read.umi.fit[
                               read.umi.fit$type=="10X_PBMC_1",],
                             gamma.mixed.fits.drop,
                             disp.fun.param.drop,
                             nCellsRange = cellsPerPerson,
                             readDepthRange = readDepth,
                             mappingEfficiency = mapping.efficiency,
```

```
                              multipletRate = multipletRate,
                              min.UMI.counts = minUMI,
                              perc.indiv.expr = perc.indiv,
                              nGenes=nFittedGenes,
                              multipletRateGrowth = multipletRateGrowth,
                              MTmethod = "FDR",
                              speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  de.raw.values.lung<-rbind(de.raw.values.lung,max.values)
}

#####
#Observed DE studies - lung (drop-seq)

de.priors.lung<-de.ref.study[de.ref.study$type=="lung",]

de.real.values.lung<-NULL
for(studyName in unique(de.priors.lung$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.libPrepCell(i,"de",ct,ct.freq,
                            libPrepCell,costFlowCell,readsPerFlowcell,
                            de.priors.lung,studyName,
                            cellsPerChanel,read.umi.fit[
                              read.umi.fit$type=="10X_PBMC_1",],
                            gamma.mixed.fits.drop,
                            disp.fun.param.drop,
                            nCellsRange = cellsPerPerson,
                            readDepthRange = readDepth,
                            mappingEfficiency = mapping.efficiency,
                            multipletRate = multipletRate,
                            min.UMI.counts = minUMI, perc.indiv.expr = perc.indiv,
                            nGenes=nFittedGenes,
                            multipletRateGrowth = multipletRateGrowth,
                            MTmethod = "FDR",
                            speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
  de.real.values.lung<-rbind(de.real.values.lung,max.values)
}
```

```
#Smart-seq parameters

#Parameters
readDepth.smart<-c(20000,50000,seq(100000,1000000,100000),1500000,2000000)
cellsPerPerson.smart<-c(50,100,200,500,700,seq(1000,9000,500))

mapping.efficiency<-0.8
min.norm.count<-minUMI
perc.indiv.expr<-perc.indiv

ct<-"alpha"
ct.freq<-0.2

prepCostCell<-13
costFlowCell<-14032
readsPerFlowcell<-4100*10^6

nFittedGenes<-22000

#Number DE genes to be modeled
numGenes<-250
```

```
#####
#Simulated DE studies - pancreas (smart-seq)

simulated.parameters<-data.frame(mean.value=c(2,0.5,2,0.5),
                                 rank.max=c(20000,20000,10000,10000),
                                 name=c("highES_unifRank","lowES_unifRank",
                                        "highES_highRank","lowES_highRank"))

print(simulated.parameters)

de.raw.values.panc<-NULL
for(s in 1:nrow(simulated.parameters)){

  print(s)

  #Simulate prior for expression rank
  ranks<-uniform.ranks.interval(start=1,end=simulated.parameters$rank.max[s],
                                numGenes=numGenes)

  #Simulate prior for effect sizes
  set.seed(1)
  effectSizes<-effectSize.DE.simulation(mean=simulated.parameters$mean.value[s],
                                        sd=1,numGenes=numGenes)

  sample.DE.genes<-data.frame(ranks=ranks, FoldChange=effectSizes,
                              geneLength=5000, name="Simulated")

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.smartseq(i,"de",ct,ct.freq,
```

```
                                        prepCostCell,costFlowCell,
                                        readsPerFlowcell,
                                        sample.DE.genes,"Simulated",
                                        gamma.mixed.fits.smart,
                                        disp.fun.param.smart,
                                        nCellsRange = cellsPerPerson.smart,
                                        readDepthRange = readDepth.smart,
                                        mappingEfficiency = mapping.efficiency,
                                        min.norm.count = min.norm.count,
                                        perc.indiv.expr = perc.indiv.expr,
                                        nGenes=nFittedGenes,
                                        MTmethod = "FDR",
                                        speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }
  max.values$budget<-budget
  max.values$parameters<-simulated.parameters$name[s]

  de.raw.values.panc<-rbind(de.raw.values.panc,max.values)
}

#####
#Observed DE studies - pancreas (smart-seq)
de.priors.panc<-de.ref.study[de.ref.study$type=="pancreas",]

de.real.values.panc<-NULL
for(studyName in unique(de.priors.panc$name)){

  max.values<-NULL
  for(i in budget){

    power.study<-optimize.constant.budget.smartseq(i,"de",ct,ct.freq,
                                    prepCostCell,costFlowCell,readsPerFlowcell,
                                    de.priors.panc,studyName,
                                    gamma.mixed.fits.smart,
                                    disp.fun.param.smart,
                                    nCellsRange = cellsPerPerson.smart,
                                    readDepthRange = readDepth.smart,
                                    mappingEfficiency = mapping.efficiency,
                                    min.norm.count = min.norm.count,
                                    perc.indiv.expr = perc.indiv.expr,
                                    nGenes=nFittedGenes,
                                    MTmethod = "FDR",
                                    speedPowerCalc=TRUE)

    max.values<-rbind(max.values,
                      power.study[which.max(power.study$powerDetect),])
  }

  max.values$budget<-budget
  max.values$parameters<-studyName
```

```
  de.real.values.panc<-rbind(de.real.values.panc,max.values)
}
```

```
#Get precalculated values
data(precalculatedBudgetOptim)
de.raw.values.lung<-budget.optimization[budget.optimization$type=="DE_simulated_lung",]
de.raw.values.panc<-budget.optimization[budget.optimization$type=="DE_simulated_pancreas",]
de.real.values.lung<-budget.optimization[budget.optimization$type=="DE_observed_lung",]
de.real.values.panc<-budget.optimization[budget.optimization$type=="DE_observed_pancreas",]

dummy.plot.smart<-data.frame(budget=c(min(budget),max(budget),min(budget),max(budget),
                                      min(budget),max(budget),min(budget),max(budget)),
                             variable=c("powerDetect","powerDetect","readDepth","readDepth",
                                        "totalCells","totalCells","sampleSize","sampleSize"),
                             value=c(0,1,0,500000,
                                     min(cellsPerPerson.smart),1000,0,100))

#DE lung simulated values
plot.values<-de.raw.values.lung[,c("budget","powerDetect","totalCells","readDepth",
                                   "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.lung<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#DE pancreas simulated values
plot.values<-de.raw.values.panc[,c("budget","powerDetect","totalCells","readDepth",
                                   "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot.smart
dummy.plot.temp$parameters<-plot.values$parameters[1]
g.panc<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Simulated parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#De lung real values
plot.values<-de.real.values.lung[,c("budget","powerDetect","totalCells","readDepth",
                                    "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot
dummy.plot.temp$parameters<-de.real.values.lung$parameters[1]
g.lung.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
```

```
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#De pancreas real values
plot.values<-de.real.values.panc[,c("budget","powerDetect","totalCells","readDepth",
                                    "sampleSize","parameters")]
plot.values<-reshape2::melt(plot.values,id.vars=c("budget","parameters"))
dummy.plot.temp<-dummy.plot.smart
dummy.plot.temp$parameters<-de.real.values.panc$parameters[1]
g.panc.2<-ggplot(plot.values,aes(x=budget,y=value,color=parameters))+
  geom_line()+geom_point()+
  facet_wrap(~variable,scales="free_y",ncol=4,labeller = labeller(variable=var.labs))+
  geom_blank(data=dummy.plot.temp)+
  scale_color_manual("Observed parameters",values=col.set2)+
  theme_bw()+
  xlab("Budget")+ylab("Value")

#Overview plot
g<-ggarrange(g.lung,g.panc,g.lung.2,g.panc.2,
             ncol=1,nrow=4,labels=c("A","B","C","D"),align="hv")
print(g)
```

Figure 5: Optimal parameters for varying budgets and Drop-seq and Smart-seq2 data. The figure shows the maximal reachable detection power (y-axis, first column) for a given experimental budget (x-axis) and

the corresponding optimal parameter combinations for that budget (y-axis, second till fourth column). The colored lines indicate different effect sizes and gene expression rank distributions. Subplots A-B visualize different simulated effect sizes and rank distributions (simulation names see text) for DE studies with models fitted on Drop-seq lung data (A) and Smart-seq2 pancreas data (B). Subplots C-D visualize effect sizes and rank distributions observed in cell type sorted bulk RNA-seq DE studies with model fits analogously to A-B.

Supplementary Figures

Figure S1: PBMC data set

Figure S1 shows the UMAP of the PBMC data set with cell type annotation and marker gene expression. Due to runtime reasons, the preprocessing is not performed here again. It is in detail described in the methods part of the publication.

Figure S2 - S4: Steps fitting the gamma mixed model

Fitting the gamma mixed model is too runtime intensive for the vignette. The fitting for a small data set is shown in the introduction vignette. For our large data set, it was done exactly the same way, using the same functions. Also the corresponding plots to Figure S2 to S4 are shown in the introduction vignette and are therefore not reproduced here again. Figure S2 shows how mean values sampled for the fitted gamma mixture distribution are compared to the observed mean values, this is shown in the introduction vignette, section "Comparison of gamma mixed fits with original means". The next section "Parameterization of the parameters of the gamma fits by the mean UMI counts per cell" shows the relationship between the gamma mixed parameters and the mean UMI counts per cell, as depicted in Figure S3. And the relationship between UMI counts and mapped read depth is shown in section "Fitting a functions for UMI counts dependent on read depth", as in Figure S4.

Figure S5: Expression probability model for a count threshold larger than zero.

The supplementary figure shows the expression probability model for all runs of our own data set. It is an extension to Figure 2, where only the results for "Run 5" are shown. For an explanation of the calculation, please look at the description text for Figure 2.

```
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

#Number of samples per run
personsPerRun<-list("Run 1"=14,"Run 2"=7, "Run 3"=7, "Run 4"=14,
                    "Run 5"=14, "Run 6"=14)

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count0",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-personsPerRun[[estimates.allSamples$run[[i]]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
```

```
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=0,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75","subsampled50",
                             "subsampled25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]

#Rename runs
plot.estimates$run<-as.character(plot.estimates$run)

g<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(fill=sample,color=sample,shape=cell.type),size=2)+
  facet_wrap(~run,ncol=2,scales="free")+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_color_manual("Target read depth",values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  scale_fill_manual(values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.position = "bottom",
        legend.direction = "horizontal", legend.box = "vertical")+
  guides(linetype=FALSE,fill=FALSE)

print(g)
```

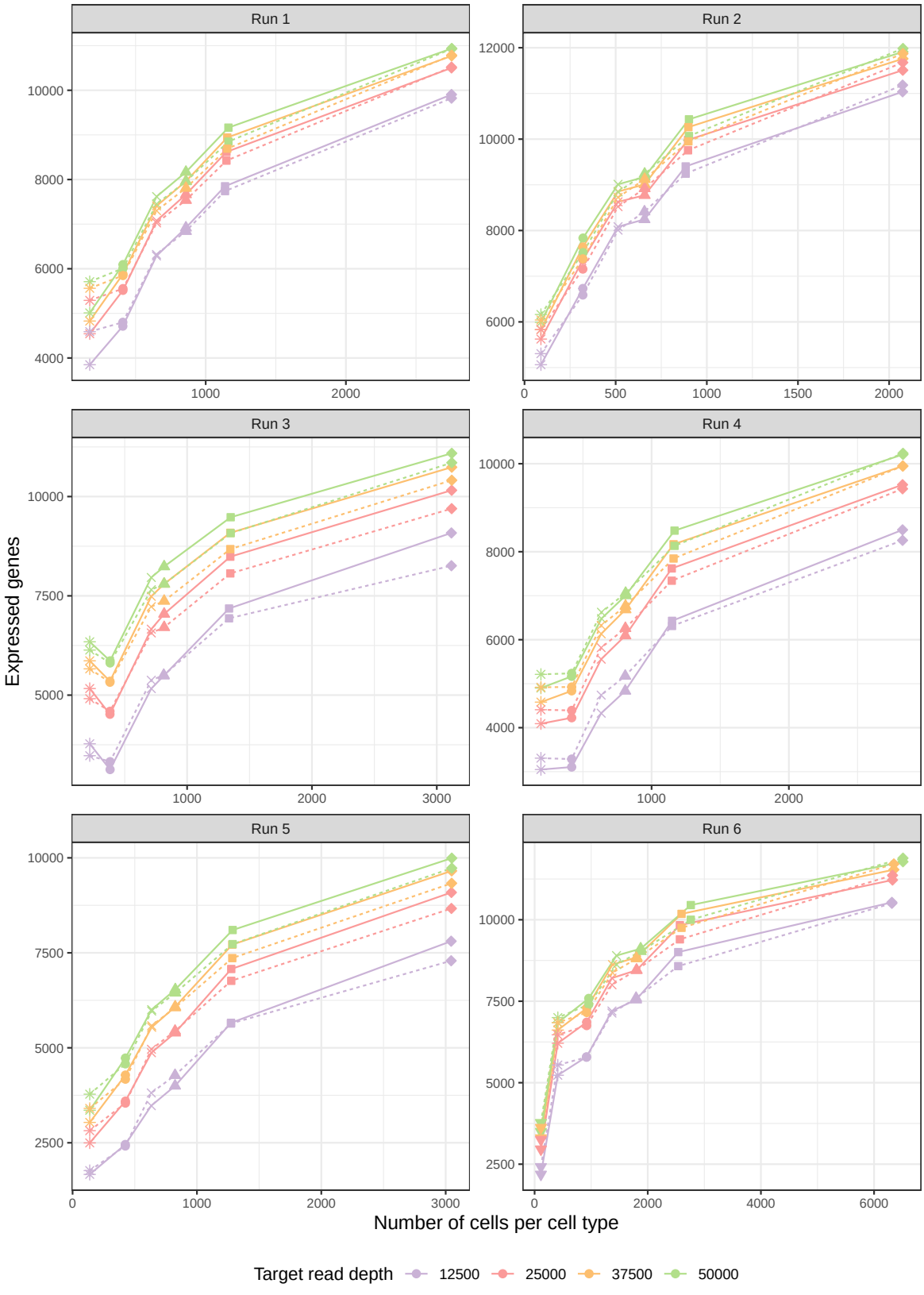
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Expressed genes
Number of cells per cell type
Target read depth
12500
25000
37500
50000
Cell type
B cells
CD14+ Monocytes
CD4 T cells
CD8 T cells
Dendritic cells
FCGR3A+ Monocytes
NK cells

Figure S5: Expression probability model for a count threshold larger than zero. Gene curve fits using our gamma mixed models parameterized over UMI counts per cell for all our six runs. The definition for an expressed gene was parameterized in the way that a gene needs to have a count > 0 in more than 50% of the individuals.

Figure S6: Expression probability model for a count threshold larger than ten.

The supplementary figure shows the same as Figure S5, but with a count threshold of ten instead of zero.

```
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

#Number of samples per run
personsPerRun<-list("Run 1"=14,"Run 2"=7, "Run 3"=7, "Run 4"=14,
                    "Run 5"=14, "Run 6"=14)

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="own_count10",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-personsPerRun[[estimates.allSamples$run[[i]]]]

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param,
    min.counts=10,
    perc.indiv=0.5)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("50000","37500","25000","12500"),
                           c("complete","subsampled75","subsampled50",
                             "subsampled25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]

#Rename runs
plot.estimates$run<-as.character(plot.estimates$run)
```

```
g<-ggplot(plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(fill=sample,color=sample,shape=cell.type),size=2)+
  facet_wrap(~run,ncol=2,scales="free")+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_color_manual("Target read depth",values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  scale_fill_manual(values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.position = "bottom",
        legend.direction = "horizontal", legend.box = "vertical")+
  guides(linetype=FALSE,fill=FALSE)

print(g)
```

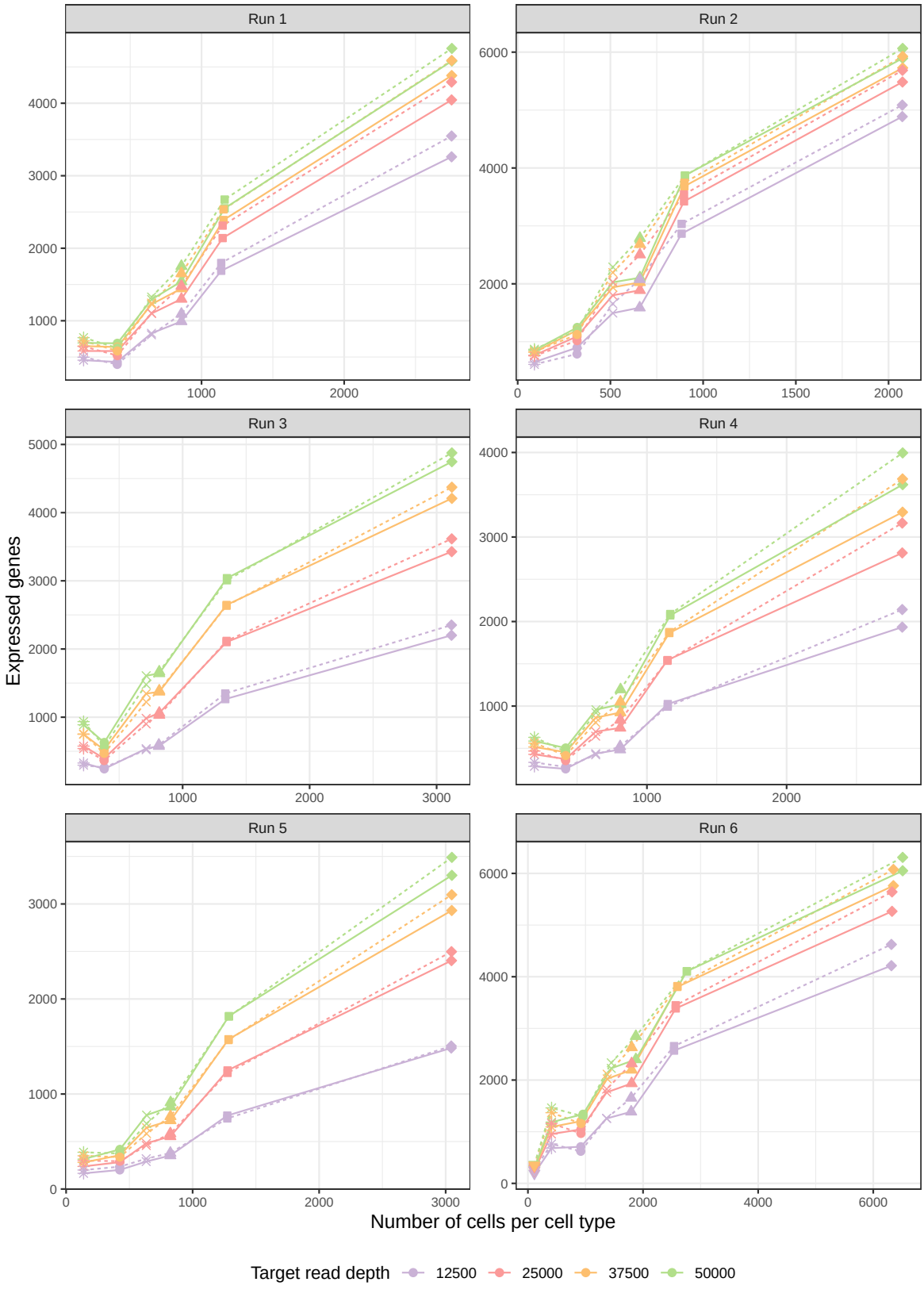
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Expressed genes
Number of cells per cell type
Target read depth
12500
25000
37500
50000
Cell type
B cells
CD14+ Monocytes
CD4 T cells
CD8 T cells
Dendritic cells
FCGR3A+ Monocytes
NK cells

Figure S6: Expression probability model for a count threshold larger than ten. Gene curve fits using our gamma mixed models parameterized over UMI counts per cell for all our six runs. The definition for an expressed gene was parameterized in the way that a gene needs to have a count > 10 in more than 50% of the individuals.

Figure S7: Influence of the number of measured cells per cell type and individual and the mean reads per cell on the number of detectable genes.

The supplemenatry figure shows how the expected number of expressed/detectable genes depends on the experimental parameters cells per person and read depth. It is an extension of Figure 3, where only the expression probability is shown.

```
ct<-"CD4 T cells"
cellCounts<-c(100,200,500,1000,1500,2000,2500,3000)
readDepths<-c(12500,25000,37500,50000)

minUMI.counts<-10
measuredIndivs<-14
numInd.threshold<-0.5
nGenes<-21000

geneCounts<-NULL
for(r in readDepths){
  for(c in cellCounts){
    count.est<-scPower::estimate.exp.prob.param(nSamples=measuredIndivs,
                                                readDepth=r,nCellsCt=c,
                                                scPower::read.umi.fit[
                                                  read.umi.fit$type=="10X_PBMC_1",],
                                                scPower::gamma.mixed.fits,
                                                ct=ct,
                                                scPower::disp.fun.param,
                                                nGenes=nGenes,
                                                min.counts=minUMI.counts,
                                                perc.indiv=numInd.threshold)

    geneCounts<-rbind(geneCounts,data.frame(meanReads=r,numCells=c,
                                            count.est=round(sum(count.est))))
  }
}

geneCounts$meanReads<-as.factor(geneCounts$meanReads)
geneCounts$numCells<-as.factor(geneCounts$numCells)
g<-ggplot(geneCounts,aes(x=numCells,y=meanReads,fill=count.est))+
  geom_tile()+
  geom_text(aes(label=count.est,color=count.est>8000))+
  scale_colour_manual(values=c("white", "black"))+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Mean reads per cell")+
  scale_fill_viridis("Detectable genes")+
  theme_bw()+
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=10),
        legend.title=element_text(size=10),
```

```
        legend.text=element_text(size=10))+
  guides(colour=FALSE)

print(g)
```

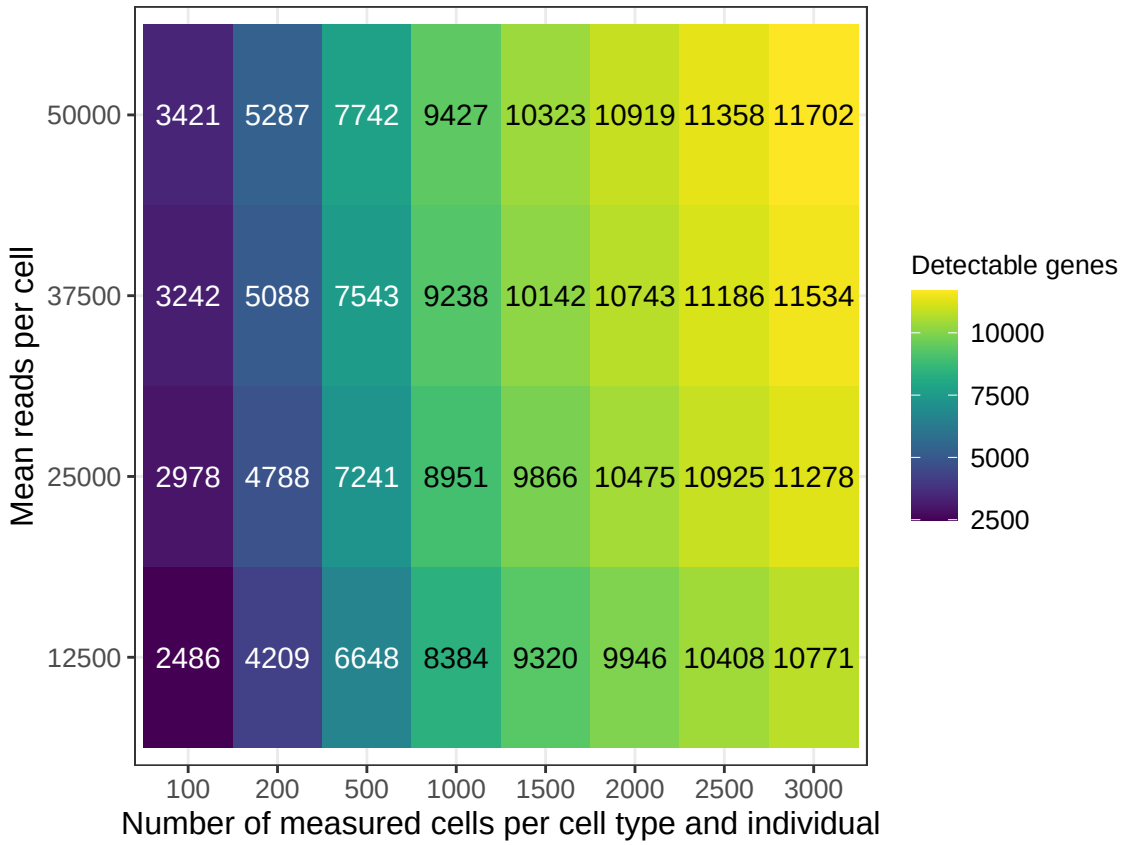

Figure S7: Influence of the number of measured cells per cell type and individual and the mean reads per cell on the number of detectable genes. Exemplarily, it is shown for the cell type CD4 T cells and a study size of 14 individuals. The definition for a detectable gene is set to at least 10 counts in more than 50% of all individuals.

Figure S8: Expression rank priors from cell type sorted bulk studies

Figure S8 shows the expression rank priors taken from all analysed DE and eQTL studies. The ranks were calculated using the function "gene.rank.calculation", as described in the introduction vignette in section "Priors from DE / eQTL studies". For citations of all used studies, please look into the publication.

```
#Load precalculated results
eqtl.ranks<-scPower::eqtl.ref.study
de.ranks<-scPower::de.ref.study[de.ref.study$type=="PBMC",]
de.ranks.lung<-scPower::de.ref.study[de.ref.study$type=="lung",]
de.ranks.panc<-scPower::de.ref.study[de.ref.study$type=="pancreas",]

#Add on the y-axis also absolute numbers (second plot)
eqtl.ranks$name<-paste0("Blueprint (",eqtl.ranks$cellType,")")
eqtl.g<-ggplot(eqtl.ranks, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of eQTL genes")+
```

```
  scale_color_manual("Study", values=col.set2[5:6])+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=2,byrow=TRUE))

de.g<-ggplot(de.ranks, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of DE genes")+
  scale_color_manual("Study", values=col.set2)+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=4,byrow=TRUE))

de.ranks.lung$name<-gsub("_"," ",de.ranks.lung$name)
de.lung<-ggplot(de.ranks.lung, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of DE genes")+
  scale_color_manual("Study", values=col.set2)+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=1,byrow=TRUE))

de.ranks.panc$name<-gsub("_"," ",de.ranks.panc$name)
de.panc<-ggplot(de.ranks.panc, aes(rank, col=name)) + stat_ecdf(geom = "step") +
  xlab("Number of expressed genes") +ylab("Percentage of DE genes")+
  scale_color_manual("Study", values=col.set2)+
  theme_bw() + xlim(0,30000) +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.position = "bottom",
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8)+
  guides(col=guide_legend(nrow=2,byrow=TRUE))

g<-ggarrange(eqtl.g,de.g,de.lung,de.panc,ncol=2,nrow=2,labels=c("A","B","C","D"),
             align="hv")
#> Warning: Removed 1 rows containing non-finite values (stat_ecdf).

print(g)
```

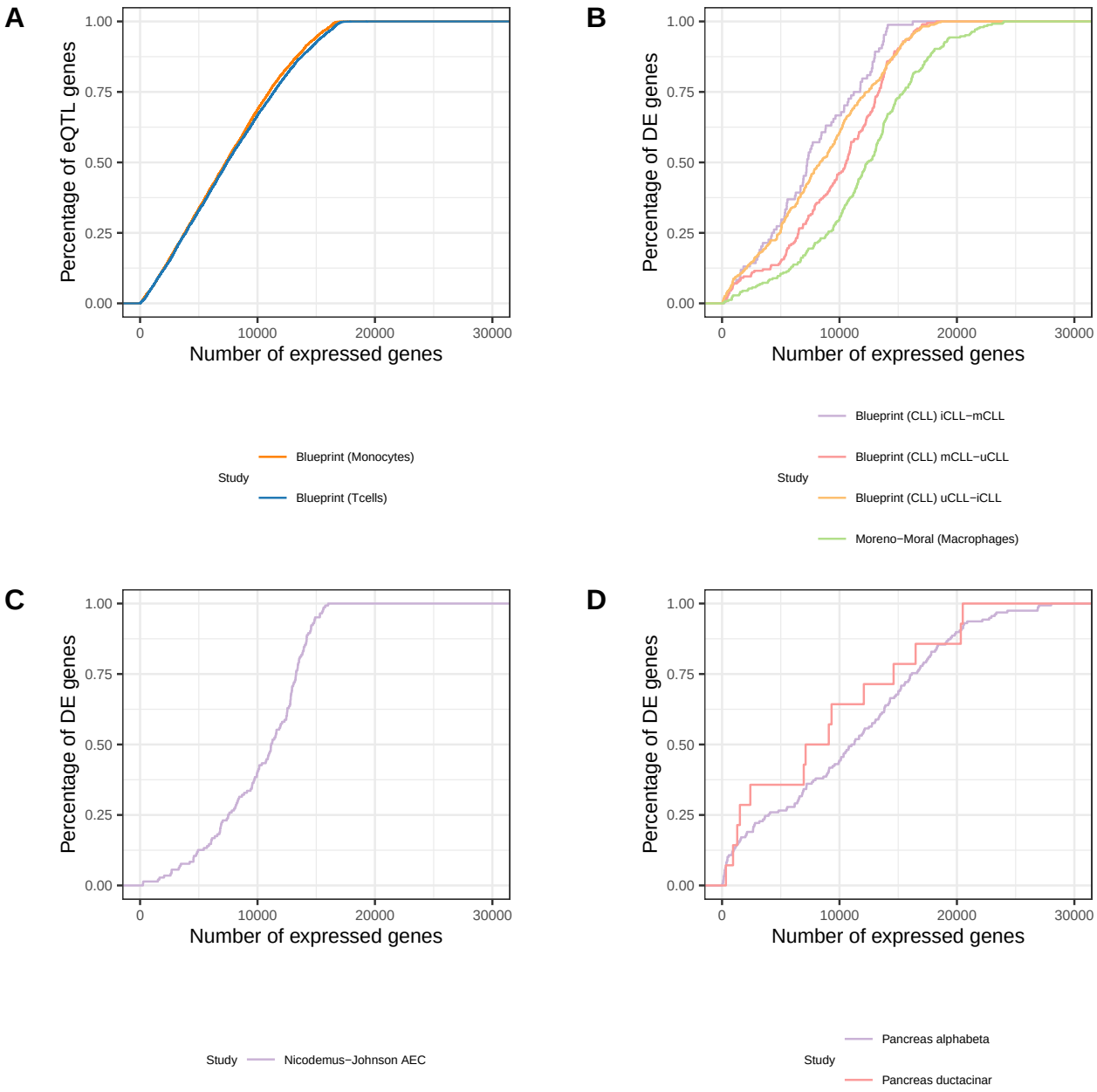

Figure S8: Expression rank priors from cell type sorted bulk studies. Gene expression distribution relative to all genes (i.e. gene expression ranks) gained from cell type sorted bulk studies for A. eQTL studies of PBMCs B. DE studies of PBMCs C. DE studies of lung tissue and D. DE studies of pancreas tissue. The figure shows which percentage of the DE/eQTL genes is expressed for a certain number of expressed genes.

Figure S9: Effect size priors from cell type sorted bulk studies

Figure S9 shows effect sizes taken directly from all analysed DE and eQTL studies, defining significant genes to be below an FDR threshold of 0.05. For citations of all used studies, please look into the publication.

```
eqtl.ranks$name<-paste0("Blueprint (",eqtl.ranks$cellType,")")
eqtl.g<-ggplot(eqtl.ranks, aes(x=beta, fill=name))+
  geom_histogram(position="dodge")+
```

```
  xlab("Beta values") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set2[5:6])+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=2,byrow=TRUE,
                           override.aes = list(size = 0.5)))

de.ranks$log2FoldChange<-log2(de.ranks$FoldChange)
de.g<-ggplot(de.ranks, aes(x=log2FoldChange, fill=name)) +
  geom_histogram(position="dodge")+
  xlab("Log fold change") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=4,byrow=TRUE,
                           override.aes = list(size = 0.5)))

de.ranks.lung$log2FoldChange<-log2(de.ranks.lung$FoldChange)
de.lung<-ggplot(de.ranks.lung, aes(x=log2FoldChange, fill=name)) +
  geom_histogram(position="dodge")+
  xlab("Log fold change") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=1,byrow=TRUE,
                           override.aes = list(size = 0.5)))

de.ranks.panc$log2FoldChange<-log2(de.ranks.panc$FoldChange)
de.panc<-ggplot(de.ranks.panc, aes(x=log2FoldChange, fill=name)) +
  geom_histogram(position="dodge")+
  xlab("Log fold change") +ylab("Counts")+
  scale_fill_manual("Study", values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=7),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        legend.position = "bottom")+
  guides(fill=guide_legend(nrow=2,byrow=TRUE,
                           override.aes = list(size = 0.5)))
```

```
g<-ggarrange(eqtl.g,de.g,de.lung,de.panc,ncol=2,nrow=2,
             labels=c("A","B","C","D"),align="hv")
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

print(g)
```

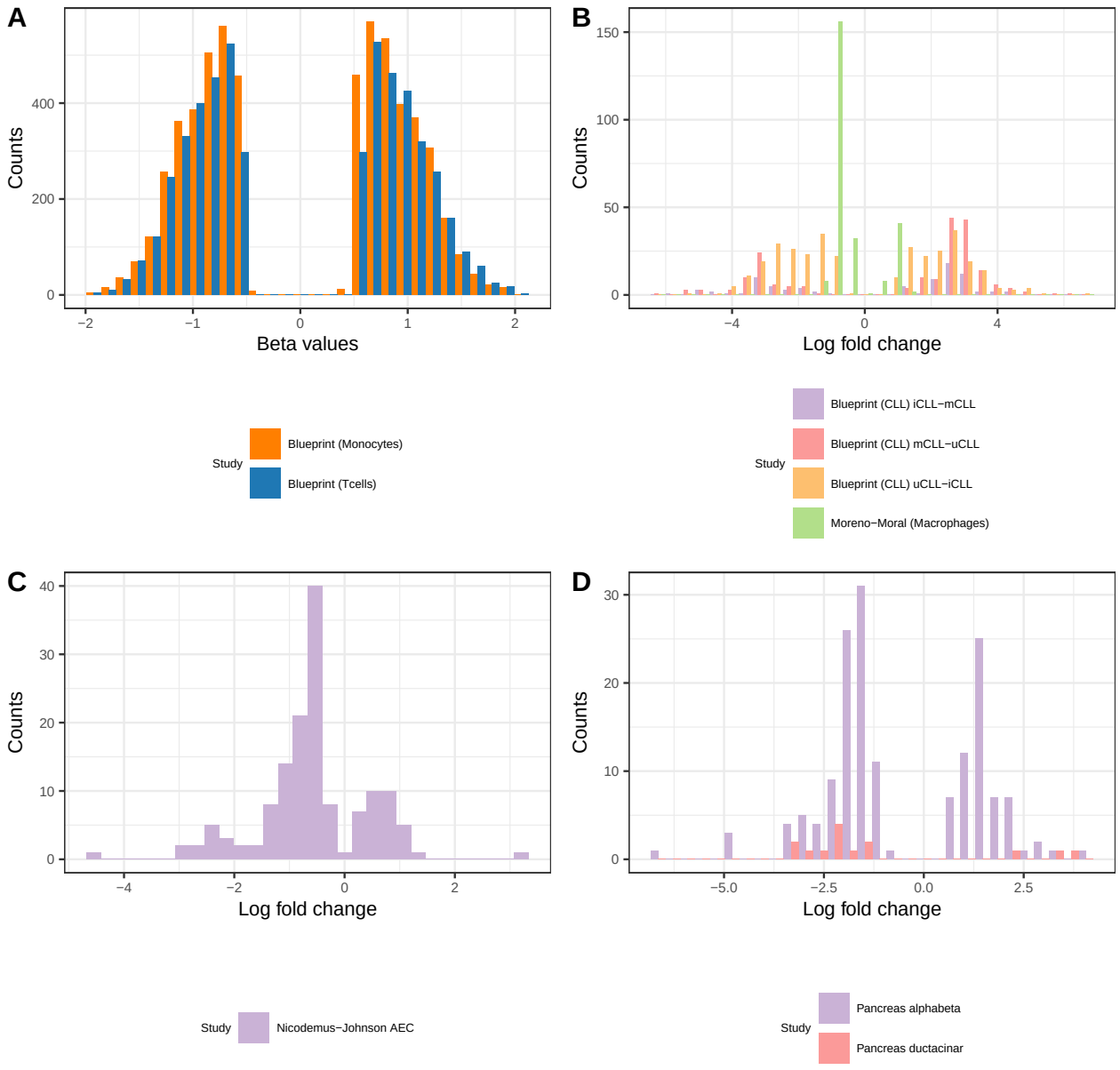

Figure S9: Effect size priors from cell type sorted bulk studies. Effect sizes gained from cell type sorted bulk studies for A. eQTL studies of PBMCs B. DE studies of PBMCs C. DE studies of lung tissue and D. DE studies of pancreas tissue. The effect size is quantified as beta values for eQTL studies and as log fold changes for DE studies.

Figure S10: Relation between eQTL power and expression mean in a simulation study

Figure S10 shows how large the deviation between simulated and analytic mean values is for small mean values, using a range of typical parameter combinations for sample size, effect size (Rsq), mean and Bonferroni adjustetd p-values.

To speed uo calculations, the plot is only shown for a reduced number of parameter combinations.

```
#Parameters used for the original plot
# count.mean<-c(seq(1,20),50,100,150)
# nGenes<-c(1000,2000,5000,10000)

#Reduced parameters used here to accelerate
count.mean<-seq(1,9)
nGenes<-1000

#Simulate look-up table of typical parameter combinations
sampleSize<-c(20,50,100,200)
Rsq<-quantile(abs(scPower::eqtl.ref.study$Rsq))

#Check also different p-values (all approximating Bonferroni corrected p-values)
pvals<-0.05/(10*nGenes)

param.combis<-expand.grid(sampleSize,Rsq,count.mean,pvals,KEEP.OUT.ATTRS = FALSE)
colnames(param.combis)<-c("sampleSize","Rsq","count.mean","pvals")

#Get simulated power
param.combis$power.simulated<-sapply(1:nrow(param.combis),function(i)
    scPower:::power.eqtl.simulated(param.combis$count.mean[i],
                                   param.combis$Rsq[i],
                                   param.combis$pvals[i],
                                   param.combis$sampleSize[i]))

#Get analytic power
param.combis$power.analytic<-sapply(1:nrow(param.combis),function(i)
    scPower:::power.eqtl.ftest(param.combis$Rsq[i],
                               param.combis$pvals[i],
                               param.combis$sampleSize[i]))

#Calculate deviation
param.combis$deviation<-param.combis$power.analytic-param.combis$power.simulated

g.a<-ggplot(param.combis[param.combis$count.mean<=20,],
            aes(x=power.simulated,y=power.analytic,color=count.mean))+
  geom_point()+geom_abline()+
  xlab("Power - eQTL simulation") + ylab("Power - scPower")+
  theme_bw()+
  scale_color_viridis("Mean")

g.b<-ggplot(param.combis,aes(x=as.factor(count.mean),y=deviation))+geom_boxplot()+
  xlab("Deviation of analytic power and simulated power")+xlab("Mean")+
  theme_bw()
```

```
g<-ggarrange(g.a,g.b,ncol=2,labels=c("A","B"))

print(g)
```

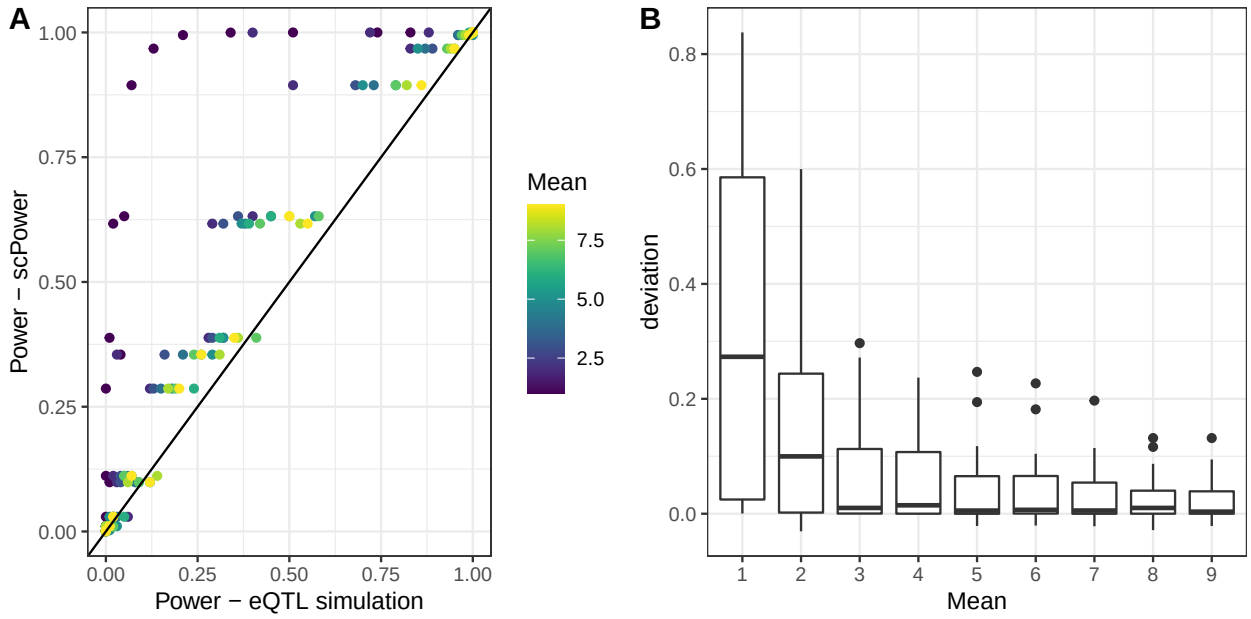

Figure S11: Detection power using observed priors from reference studies

Figure S11 is an extension of Figure 3, showing the detection power for all analysed reference studies with PBMC cell types and matched cell type expression model, while in Figure 3 it is only shown for one DE and one eQTL study.

In contrast to the original publication, the parameter useSimulatedPower is set to FALSE for the eQTL budget optimization. Both parameters speed the calculation in the vignette, while changing the overall power only slightly. We show in the publication the more accurate version including simulated eQTL power for lowly expressed genes.

```
#General parameters
transcriptome.mapped.reads<-20000
nGenes<-21000
ct.freq<-1
cellsPerLane<-20000

########
# DE subplot
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(4,6,8,10,16,20)

#Build a frame of all possible combinations and reference studies
param.combis<-expand.grid(cellsPerCelltype,sampleSize,
                          unique(de.ref.study$name[de.ref.study$type=="PBMC"]))
colnames(param.combis)<-c("cellsPerCelltype","sampleSize","refStudyName")

#Match for each study the corresponding cell type
matched.ct<-data.frame(refStudyName=unique(de.ref.study$name[de.ref.study$type==
```

```
                                                                "PBMC"]),
                      ct=c(rep("CD4 T cells",3),"CD14+ Monocytes"),
                      stringsAsFactors=FALSE)
print(matched.ct)
#>                  refStudyName              ct
#> 1  Blueprint (CLL) iCLL-mCLL     CD4 T cells
#> 2  Blueprint (CLL) mCLL-uCLL     CD4 T cells
#> 3  Blueprint (CLL) uCLL-iCLL     CD4 T cells
#> 4 Moreno-Moral (Macrophages) CD14+ Monocytes
param.combis<-merge(param.combis,matched.ct,by="refStudyName")
param.combis$refStudyName<-as.character(param.combis$refStudyName)

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                                   param.combis$cellsPerCelltype[i],
                                   transcriptome.mapped.reads,
                                   ct.freq, "de",de.ref.study,
                                   param.combis$refStudyName[i],
                                   cellsPerLane,read.umi.fit[
                                     read.umi.fit$type=="10X_PBMC_1",],
                                   gamma.mixed.fits,
                                   param.combis$ct[i],
                                   disp.fun.param,mappingEfficiency = 1,
                                   multipletRate=0,multipletFactor=1,
                                   min.UMI.counts = 10,
                                   perc.indiv.expr = 0.5,
                                   nGenes=nGenes,
                                   sign.threshold = 0.05,
                                   MTmethod= "FDR"))

power.DE.study<-rbindlist(res)

#Convert axes to factors
power.DE.study$sampleSize<-as.factor(power.DE.study$sampleSize)
power.DE.study$totalCells<-as.factor(power.DE.study$totalCells)

g.DE<-ggplot(power.DE.study,aes(x=totalCells,y=sampleSize,fill=powerDetect))+
  geom_tile()+facet_wrap(~name,ncol=2)+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Detection power",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

##########
#eQTL subplot

cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(50,100,150,200)
```

```r
#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize,unique(eqtl.ref.study$name))
colnames(param.combis)<-c("cellsPerCelltype","sampleSize","refStudyName")

#Match for each study the corresponding cell type
matched.ct<-data.frame(refStudyName=unique(eqtl.ref.study$name),
                       ct=c("CD14+ Monocytes","CD4 T cells"),
                       stringsAsFactors=FALSE)
print(matched.ct)
#>            refStudyName             ct
#> 1 Blueprint (Monocytes) CD14+ Monocytes
#> 2    Blueprint (Tcells)     CD4 T cells
param.combis<-merge(param.combis,matched.ct,by="refStudyName")
param.combis$refStudyName<-as.character(param.combis$refStudyName)

#Test results for each parameter combination
res<-lapply(1:nrow(param.combis),function(i)
  power.general.restrictedDoublets(param.combis$sampleSize[i],
                                   param.combis$cellsPerCelltype[i],
                                   transcriptome.mapped.reads,
                                   ct.freq, "eqtl",eqtl.ref.study,
                                   param.combis$refStudyName[i],
                                   cellsPerLane,read.umi.fit[
                                     read.umi.fit$type=="10X_PBMC_1",],
                                   gamma.mixed.fits,
                                   param.combis$ct[i],
                                   disp.fun.param,mappingEfficiency = 1,
                                   multipletRate=0,multipletFactor=1,
                                   min.UMI.counts = 10,
                                   perc.indiv.expr = 0.5,
                                   nGenes=21000,
                                   sign.threshold = 0.05,
                                   MTmethod= "Bonferroni",
                                   #The parameter is set differently
                                   #in the publictions, but changed here to
                                   #speed the calculation
                                   useSimulatedPower = FALSE)) #(in publication: TRUE)

power.eqtl.study<-rbindlist(res)

#Convert axes to factors
power.eqtl.study$sampleSize<-as.factor(power.eqtl.study$sampleSize)
power.eqtl.study$totalCells<-as.factor(power.eqtl.study$totalCells)

g.eQTL<-ggplot(power.eqtl.study,aes(x=totalCells,y=sampleSize,fill=powerDetect))+
  geom_tile()+facet_wrap(~name,ncol=2)+
  xlab("Number of measured cells per cell type and individual")+
  ylab("Total sample size")+
  scale_fill_viridis("Detection power",limits=c(0,1))+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
```

```
        legend.text=element_text(size=8))

g<-ggarrange(g.DE,g.eQTL,ncol=1,nrow=2,heights=c(2,1),labels=c("A","B"),
          align="hv",common.legend = TRUE,legend="right")
#> Warning: Graphs cannot be horizontally aligned unless the axis parameter is set.
#> Placing graphs unaligned.

print(g)
```

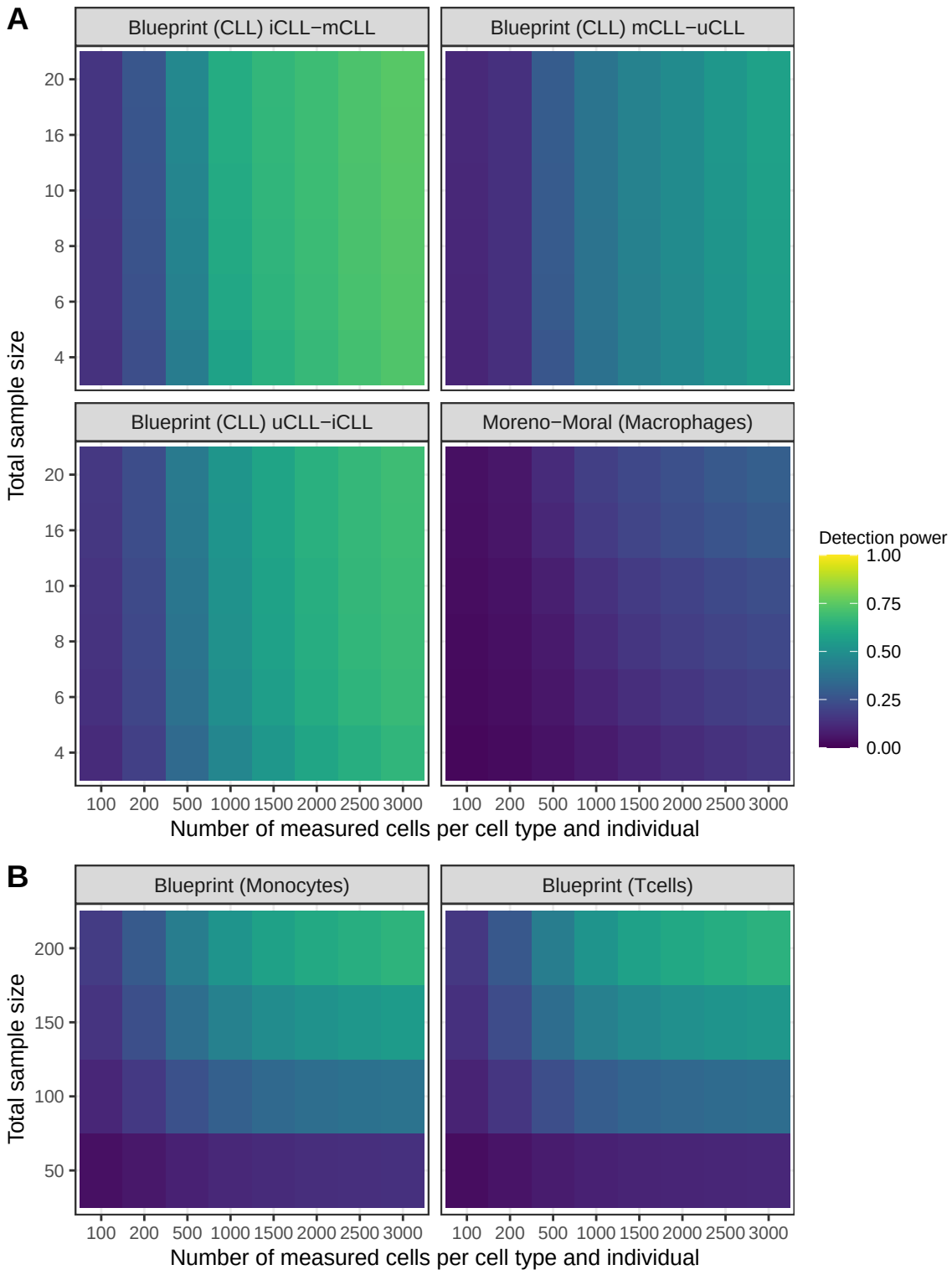

Figure S11: Detection power using observed priors from reference studies. Detection power for A. DE genes and B. eQTL genes dependent on the study, total sample size and the number of measured cells per cell type for a transcriptome mapped read depth per cell of 20,000. The detection power is the product of the probability that the gene is expressed and the power to detect it as a DE or eQTL gene, respectively, assuming that it is expressed. The fold change for DE genes and the R^2 for eQTL genes is taken from the published study, together with the expression rank of the genes. The expression profile in a single cell experiment with

a specific number of samples and measured cells is estimated using our gamma mixed models.

Figure S12: Variant of Main Figure 3 with FWER adjusted significance thresholds

This figure was generated with exactly the same code as Main Figures 3, only setting the multiple testing method (MTmethod) to "FWER" instead of "FDR" for the DE power.

Figure S13: Comparison of powsimR in combination with different DE methods and scPower

```
#General parameters
transcriptome.mapped.reads<-25000
nGenes<-21000
ct<-"CD4 T cells"
ct.freq<-1
cellsPerLane<-20000
cellsPerCelltype<-c(200,1000,3000)
sampleSize<-c(4,8,16)
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"

#Build a frame of all possible combinations
param.combis<-expand.grid(cellsPerCelltype,sampleSize,KEEP.OUT.ATTRS = FALSE)
colnames(param.combis)<-c("cellsPerCelltype","sampleSize")

#Filter ref study for the ones that are kept after gene filtering
ref.study<-de.ref.study[de.ref.study$name==ref.study.name,]
fitted.genes<-16197 #Hard-coded here, found in powsimR object

##############################################################################
#Test results for each parameter combination with MT method FWER

overview.results<-NULL
for(i in 1:nrow(param.combis)){
  detailed<-power.general.restrictedDoublets(param.combis$sampleSize[i],
                              param.combis$cellsPerCelltype[i],
                              transcriptome.mapped.reads,
                              ct.freq, "de",de.ref.study,ref.study.name,
                              cellsPerLane,read.umi.fit[
                                read.umi.fit$type=="10X_PBMC_1",],
                              gamma.mixed.fits,ct,
                              disp.fun.param,mappingEfficiency = 1,
                              multipletRate=0,multipletFactor=1,
                              min.UMI.counts = 0,
                              perc.indiv.expr = 0,
                              nGenes=21000,
                              returnResultsDetailed = TRUE,
                              MTmethod="Bonferroni")

  temp.overview<-detailed$overview.df
```

```
  #Filter power for the expressed genes
  ref.study$not.filtered<-ref.study$rank<=min(fitted.genes,
                                              detailed$overview.df$expressedGenes)
  temp.overview$power.filtered<-mean(detailed$probs.df$power[ref.study$not.filtered])

  #Filter overall power for simulated genes
  ref.study$not.filtered<-ref.study$rank<=fitted.genes
  temp.overview$overall.power.filtered<-mean(detailed$probs.df$combined.prob[
    ref.study$not.filtered])

  overview.results<-rbind(overview.results,temp.overview)
}

overview.results$method<-"scPower"
overview.results$power.sd<-0
overview.results$overall.power.sd<-0
overview.results$expressed.genes.sd<-0
overview.results<-overview.results[,c("sampleSize","totalCells","method",
                                      "power.filtered","power.sd",
                                      "overall.power.filtered","overall.power.sd",
                                      "expressedGenes","expressed.genes.sd")]
overview.results$MT<-"FWER"

#Bonferroni corrected values
power.fwer<-scPower::simulated.DE.values[simulated.DE.values$MT=="FWER",]
colnames(overview.results)<-colnames(power.fwer)
power.fwer<-rbind(power.fwer,overview.results)
power.fwer$totalCells<-as.factor(power.fwer$totalCells)

g.a<-ggplot(power.fwer,aes(x=totalCells,y=expressed.genes,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=expressed.genes-expressed.genes.sd,
                    ymax=expressed.genes+expressed.genes.sd), width=.2,
                position=position_dodge(.9))+
  ylab("Expressed genes")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+
  theme_bw()+
  scale_fill_manual("Method",values=col.set2)

g.b<-ggplot(data=power.fwer,aes(x=totalCells,y=power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=power-power.sd, ymax=power+power.sd), width=.2,
                position=position_dodge(.9))+
  ylab("DE power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=col.set2)

g.c<-ggplot(data=power.fwer,aes(x=totalCells,y=overall.power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=overall.power-overall.power.sd,
                    ymax=overall.power+overall.power.sd), width=.2,
                position=position_dodge(.9))+
```

```
  ylab("Overall detection power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=col.set2)

##############################################################################
#Test results for each parameter combination with MT method FDR

overview.results<-NULL
for(i in 1:nrow(param.combis)){
  detailed<-power.general.restrictedDoublets(param.combis$sampleSize[i],
                              param.combis$cellsPerCelltype[i],
                              transcriptome.mapped.reads,
                              ct.freq, "de",de.ref.study,ref.study.name,
                              cellsPerLane,read.umi.fit[
                                read.umi.fit$type=="10X_PBMC_1",],
                              gamma.mixed.fits,ct,
                              disp.fun.param,mappingEfficiency = 1,
                              multipletRate=0,multipletFactor=1,
                              min.UMI.counts = 0,
                              perc.indiv.expr = 0,
                              nGenes=21000,
                              returnResultsDetailed = TRUE,
                              MTmethod="FDR")

  temp.overview<-detailed$overview.df

  #Filter power for the expressed genes
  ref.study$not.filtered<-ref.study$rank<=min(fitted.genes,
                                              detailed$overview.df$expressedGenes)
  temp.overview$power.filtered<-mean(detailed$probs.df$power[ref.study$not.filtered])

  #Filter overall power for simulated genes
  ref.study$not.filtered<-ref.study$rank<=fitted.genes
  temp.overview$overall.power.filtered<-mean(detailed$probs.df$combined.prob[
    ref.study$not.filtered])

  overview.results<-rbind(overview.results,temp.overview)
}

overview.results$method<-"scPower"
overview.results$power.sd<-0
overview.results$overall.power.sd<-0
overview.results$expressed.genes.sd<-0
overview.results<-overview.results[,c("sampleSize","totalCells","method",
                                      "power.filtered","power.sd",
                                      "overall.power.filtered","overall.power.sd",
                                      "expressedGenes","expressed.genes.sd")]
overview.results$MT<-"FDR"

#Bonferroni corrected values
power.fdr<-scPower::simulated.DE.values[simulated.DE.values$MT=="FDR",]
colnames(overview.results)<-colnames(power.fdr)
```

```
power.fdr<-rbind(power.fdr,overview.results)
power.fdr$totalCells<-as.factor(power.fdr$totalCells)

g.d<-ggplot(power.fdr,aes(x=totalCells,y=expressed.genes,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=expressed.genes-expressed.genes.sd,
                    ymax=expressed.genes+expressed.genes.sd), width=.2,
                position=position_dodge(.9))+
  ylab("Expressed genes")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+
  theme_bw()+
  scale_fill_manual("Method",values=col.set2)

g.e<-ggplot(data=power.fdr,aes(x=totalCells,y=power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=power-power.sd, ymax=power+power.sd), width=.2,
                position=position_dodge(.9))+
  ylab("DE power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=col.set2)

g.f<-ggplot(data=power.fdr,aes(x=totalCells,y=overall.power,fill=method))+
  geom_bar(stat="identity",position="dodge")+
  geom_errorbar(aes(ymin=overall.power-overall.power.sd,
                    ymax=overall.power+overall.power.sd), width=.2,
                position=position_dodge(.9))+
  ylab("Overall detection power")+xlab("Cells per person")+
  facet_wrap(~sampleSize)+ylim(0,1)+
  theme_bw()+
  scale_fill_manual("Method",values=col.set2)

g<-ggarrange(g.a,g.d,
             g.b,g.e,
             g.c,g.f,
             ncol=2,nrow=3,
             labels=c("A","B","C","D","E","F"),
             common.legend=TRUE,
             legend="bottom",align="hv")

print(g)
```

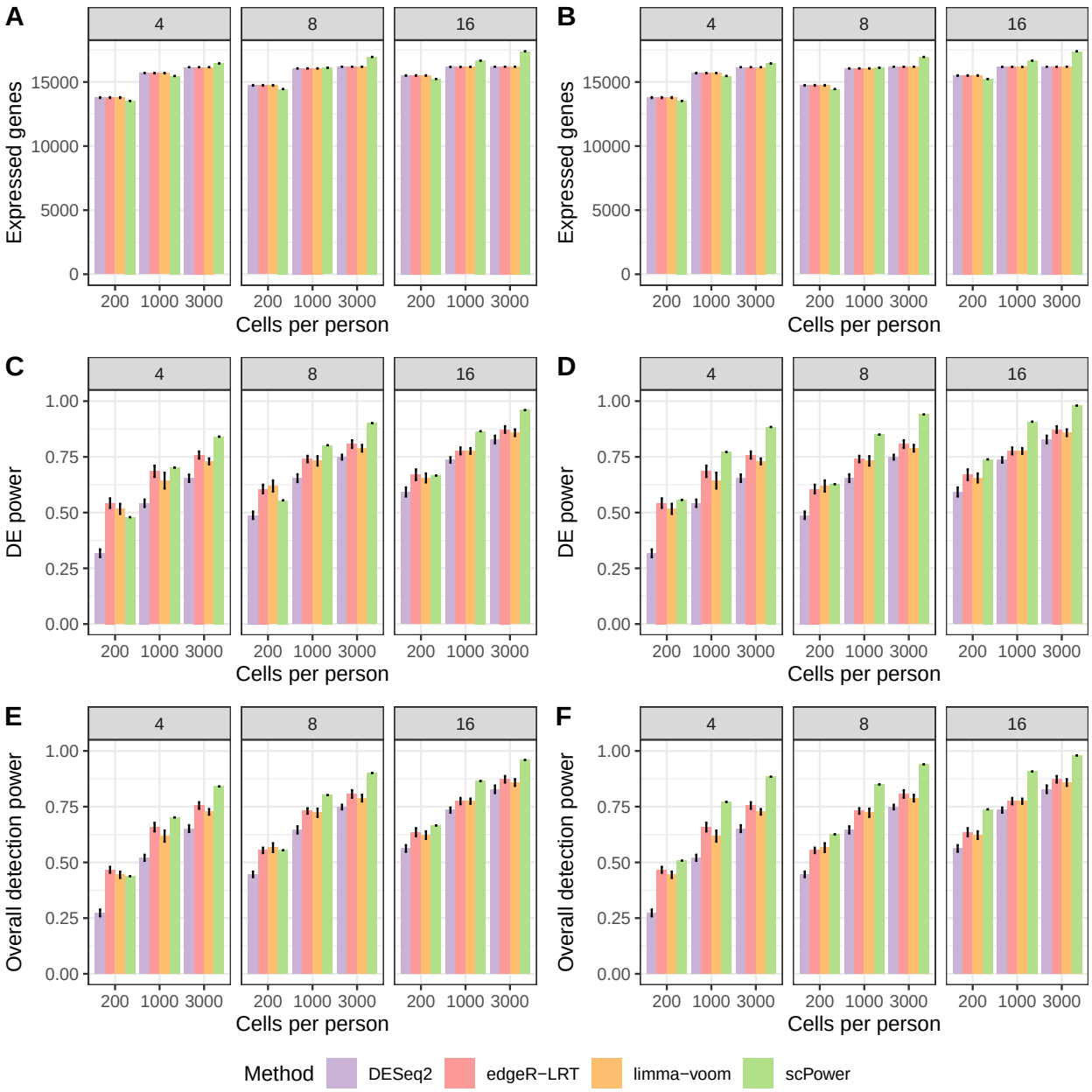

Figure S14: Comparison between powsimR and scPower over a large range of experimental design

Figure S14 extends the comparison between powsimR and scPower for a large range of parameter combinations and only evaluating it for powsimR in combination with edgeR. The used parameter combinations are:

```
cellsPerCelltype<-c(100,200,500,1000,1500,2000,2500,3000)
sampleSize<-c(4,6,8,10,16,20)
```

The final plots (restricted for the parameter combinations generated in Figure S13) are generated as:

```
#Create plot showing reproduction of main figure 3

#FWER corrected threshold
```

```
power.fwer.edgeR<-power.fwer[power.fwer$method == "edgeR-LRT",]
power.fwer.scpower<-power.fwer[power.fwer$method == "scPower",]
power.fwer<-merge(power.fwer.edgeR,power.fwer.scpower,
                  by=c("sampleSize","totalCells"),
                  suffixes=c("",".scpower"))

g.a<-ggplot(power.fwer,aes(x=expressed.genes,y=expressed.genes.scpower))+
  geom_point()+geom_abline()+geom_hline(yintercept = 16171)+
  xlab("Expressed genes - powsimR")+ylab("Expressed genes - scPower")+
  theme_bw()

g.b<-ggplot(power.fwer,aes(x=power,y=power.scpower))+
  geom_point()+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  xlab("DE power - powsimR")+ylab("DE power - scPower")+
  theme_bw()

g.c<-ggplot(power.fwer,aes(x=overall.power,y=overall.power.scpower))+
  geom_point()+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  xlab("Overall power - powsimR")+
  ylab("Overall power - scPower")+
  theme_bw()

#FDR corrected thresold
power.fdr.edgeR<-power.fdr[power.fdr$method == "edgeR-LRT",]
power.fdr.scpower<-power.fdr[power.fdr$method == "scPower",]
power.fdr<-merge(power.fdr.edgeR,power.fdr.scpower,
                  by=c("sampleSize","totalCells"),
                  suffixes=c("",".scpower"))

g.d<-ggplot(power.fdr,aes(x=expressed.genes,y=expressed.genes.scpower))+
  geom_point()+geom_abline()+geom_hline(yintercept = 16171)+
  xlab("Expressed genes - powsimR")+ylab("Expressed genes - scPower")+
  theme_bw()

g.e<-ggplot(power.fdr,aes(x=power,y=power.scpower))+
  geom_point()+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  xlab("DE power - powsimR")+ylab("DE power - scPower")+
  theme_bw()

g.f<-ggplot(power.fdr,aes(x=overall.power,y=overall.power.scpower))+
  geom_point()+geom_abline()+ylim(c(0,1))+xlim(c(0,1))+
  xlab("Overall power - powsimR")+
  ylab("Overall power - scPower")+
  theme_bw()

g<-ggarrange(g.a,g.b,g.c,
             g.d,g.e,g.f,
             ncol=3,nrow=2,
             labels=c("A","B","C","D","E","F"),align="hv")

print(g)
```

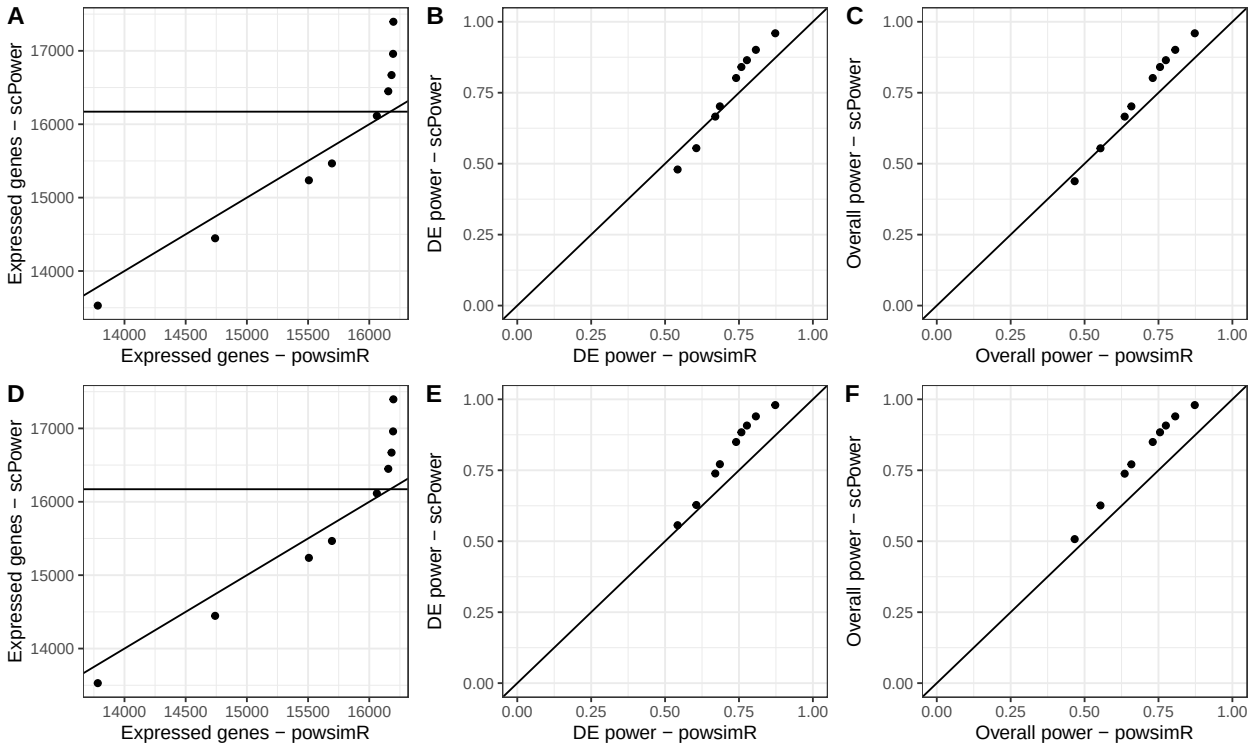

Figure S15: Parameter optimization for constant budget

Figure S15 shows two example calculations on how scPower selects the optimal experimental parameter combination for a specific budget, once for a DE study and once for an eQTL study. For an explanation of all the used parameters, please look into the introduction vignette.

In contrast to the original publication, the parameter useSimulatedPower is set to FALSE and the parameter speedPowerCalc to TRUE for the eQTL budget optimization. Both parameters speed the calculation in the vignette, while changing the overall power only slightly. We show in the publication the more accurate version including simulated eQTL power for lowly expressed genes.

```
costKit<-5600
costFlowCell<-14032
readsPerFlowcell<-4100*10^6
cellsPerLane<-20000
ct<-"CD4 T cells"
ct.freq<-0.25
mappingEfficiency<-0.8
########
# DE subplot

comp.type<-"de"
ref.study.name<-"Blueprint (CLL) iCLL-mCLL"
readDepth<-c(2000,5000,10000,15000,20000,30000,50000,70000)
cellsPerIndividual<-c(100,200,500,700,seq(1000,12000,500))
totalBudget<-10000

power.study.de<-optimize.constant.budget.restrictedDoublets(totalBudget,
                                        comp.type,ct,ct.freq,
```

```
                                    costKit,costFlowCell,readsPerFlowcell,
                                    de.ref.study,ref.study.name,
                                    cellsPerLane,
                                    read.umi.fit[
                                      read.umi.fit$type=="10X_PBMC_1",],
                                    gamma.mixed.fits,
                                    disp.fun.param,
                                    nCellsRange=cellsPerIndividual,
                                    readDepthRange=readDepth,
                                    mappingEfficiency=mappingEfficiency,
                                    sign.threshold = 0.05,
                                    MTmethod="FDR")

#Print maximal combination of values
max.study<-power.study.de[which.max(power.study.de$powerDetect),]

#Replace subsampling
colnames(power.study.de)[2:4]<-c("Detection power","Expression probability",
                                 "DE power")

#Show two-dimensional plots with always one parameter fixed
#Fixed read depths
power.study.plot<-power.study.de[power.study.de$readDepth==max.study$readDepth,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","DE power",
                                           "Detection power"))

g.rd<-ggplot(power.study.plot,aes(totalCells,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.rd)$layout$panel_params[[1]]$x.labels)

g.rd<-g.rd +
  scale_x_continuous("Cells per individual",sec.axis=sec_axis(~., breaks=labels,
                                            labels=sapply(labels,function(c)floor(
                                                          sampleSizeBudgetCalculation.restri
    totalBudget, costKit, cellsPerLane, costFlowCell,readsPerFlowcell))),
    name="Sample size"))+
  geom_vline(xintercept = max.study$totalCells)+
  ylab("Probability")+
  scale_color_manual("",values=col.set)+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

#Fixed number of cells
```

```
power.study.plot<-power.study.de[power.study.de$totalCells==max.study$totalCells,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","DE power",
                                           "Detection power"))

g.cp<-ggplot(power.study.plot,aes(readDepth,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.cp)$layout$panel_params[[1]]$x.labels)

g.cp <- g.cp + scale_x_continuous("Read depth",sec.axis=sec_axis(~., breaks=labels,
                                                     labels=sapply(labels,function(c)floor(
                                                       sampleSizeBudgetCalculation.restrictedDoublets(r
    c,totalBudget, costKit, cellsPerLane, costFlowCell,readsPerFlowcell))),
  geom_vline(xintercept = max.study$readDepth)+
  ylab("Probability")+
  scale_color_manual("",values=col.set)+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

g.a<-ggarrange(g.rd,g.cp,ncol=2,nrow=1,labels=c("A","B"),
          align="hv",common.legend=TRUE,legend="bottom")
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf

########
# eQTL subplot

comp.type<-"eqtl"
ref.study.name<-"Blueprint (Tcells)"
readDepth<-c(2000,5000,10000,15000,20000,30000,50000)
cellsPerIndividual<-c(100,200,500,700,seq(1000,9000,500))
totalBudget<-30000

power.study.eqtl<-optimize.constant.budget.restrictedDoublets(totalBudget,
```

```
                                    comp.type,ct,ct.freq,
                                    costKit,costFlowCell,readsPerFlowcell,
                                    eqtl.ref.study,ref.study.name,
                                    cellsPerLane,
                                    read.umi.fit[
                                      read.umi.fit$type=="10X_PBMC_1",],
                                    gamma.mixed.fits,
                                    disp.fun.param,
                                    nCellsRange=cellsPerIndividual,
                                    readDepthRange=readDepth,
                                    mappingEfficiency=mappingEfficiency,
                                    sign.threshold = 0.05,
                                    MTmethod="Bonferroni",
                                    #These two parameters are set differently
                                    #in the publictions, but changed here to
                                    #speed the calculation
                                    useSimulatedPower = FALSE, #(in publication: TRUE)
                                    speedPowerCalc = TRUE) #(in publication: FALSE)

#Print maximal combination of values
max.study<-power.study.eqtl[which.max(power.study.eqtl$powerDetect),]

#Replace subsampling
colnames(power.study.eqtl)[2:4]<-c("Detection power","Expression probability",
                                   "eQTL power")

#Show two-dimensional plots with always one parameter fixed
#Fixed read depths
power.study.plot<-power.study.eqtl[power.study.eqtl$readDepth==max.study$readDepth,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","eQTL power",
                                           "Detection power"))

g.rd<-ggplot(power.study.plot,aes(totalCells,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.rd)$layout$panel_params[[1]]$x.labels)

g.rd<-g.rd +
  scale_x_continuous("Cells per individual",sec.axis=sec_axis(~., breaks=labels,
                                                              labels=sapply(labels,function(c)floor(sam
                                                                max.study$totalCells,
    c,totalBudget, costKit, cellsPerLane, costFlowCell,readsPerFlowcell))),
  geom_vline(xintercept = max.study$totalCells)+
  ylab("Probability")+
  scale_color_manual("",values=col.set)+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
```

```
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

#Fixed number of cells
power.study.plot<-power.study.eqtl[power.study.eqtl$totalCells==max.study$totalCells,]
power.study.plot<-reshape2::melt(power.study.plot,
                                 id.vars=c("name","sampleSize","totalCells",
                                           "usableCells","multipletFraction",
                                           "ctCells","readDepth","readDepthSinglet",
                                           "mappedReadDepth","expressedGenes"))
power.study.plot$variable<-factor(power.study.plot$variable,
                                  levels=c("Expression probability","eQTL power",
                                           "Detection power"))

g.cp<-ggplot(power.study.plot,aes(readDepth,value,color=variable))+geom_line()
labels<-as.numeric(ggplot_build(g.cp)$layout$panel_params[[1]]$x.labels)

g.cp <- g.cp + scale_x_continuous("Read depth",sec.axis=sec_axis(~., breaks=labels,
                                                                 labels=sapply(labels,function(c)floor(sampleSizeBu
                                                                   max.study$totalCells,
    c,totalBudget, costKit, cellsPerLane, costFlowCell,readsPerFlowcell))),
  geom_vline(xintercept = max.study$readDepth)+
  scale_color_manual("",values=col.set)+
  ylab("Probability")+
  ylim(0,1)+
  theme_bw() +
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.position=c(0.85,0.3),
        legend.title=element_blank(),
        legend.text=element_text(size=8))

g.b<-ggarrange(g.rd,g.cp,ncol=2,nrow=1,labels=c("C","D"),
          align="hv",common.legend=TRUE,legend="bottom")
#> Warning in min(x): no non-missing arguments to min; returning Inf

#> Warning in min(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf
#> Warning in min(x): no non-missing arguments to min; returning Inf
#> Warning in max(x): no non-missing arguments to max; returning -Inf

g<-ggarrange(g.a,g.b,ncol=1,nrow=2)

print(g)
```

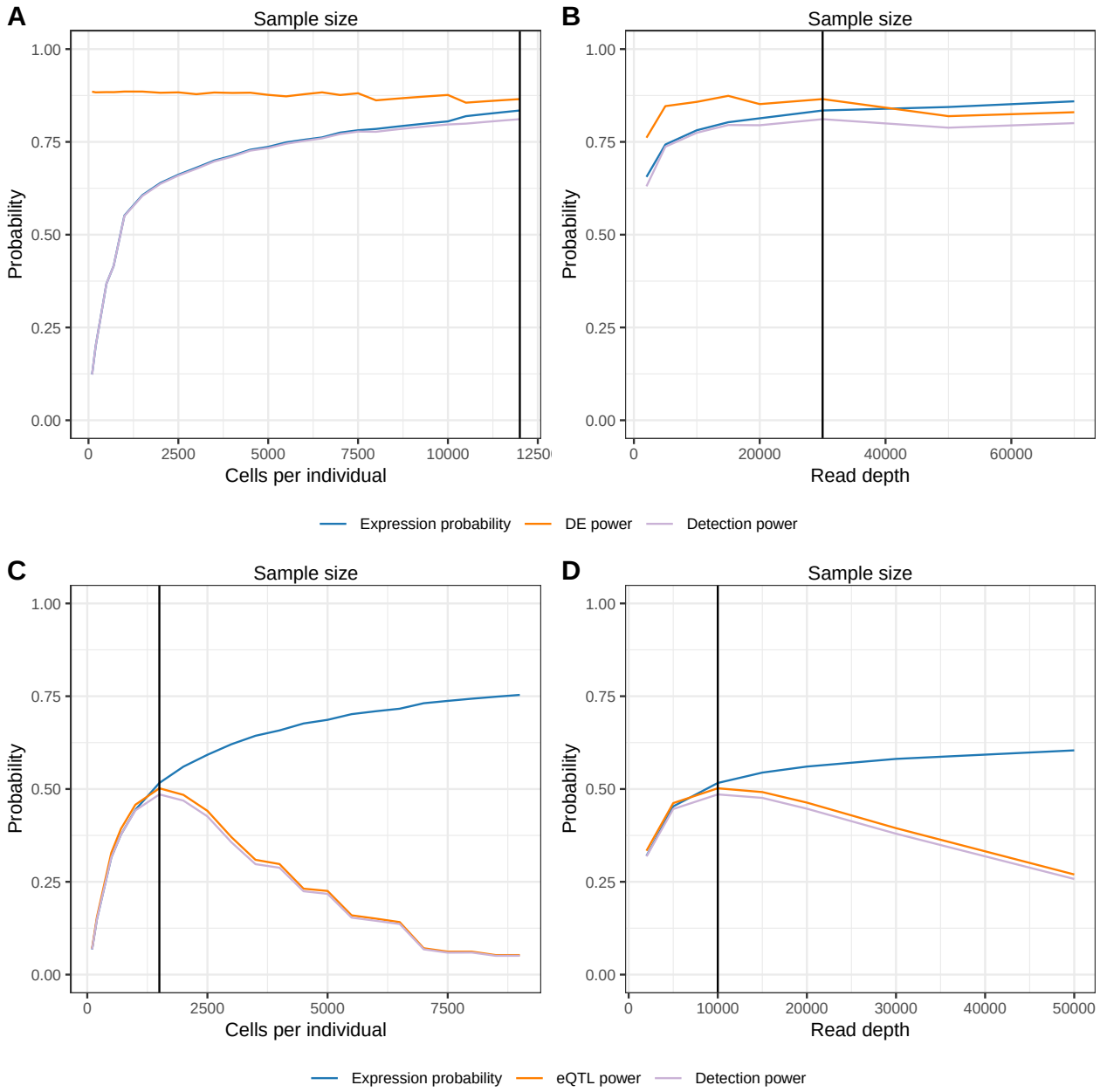

Figure S15: Parameter optimization for constant budget. Optimized parameters (sample size, read depth and cells per individual) to maximize the detection power for a DE study with a budget of 10,000€ (A,B) and an eQTL study with a budget of 30,000€ (D,E), both for a cell type with a frequency of 25%. Suplots A and C show the influence of the cells per individual given the optimized read depth, subplots B and D show the influence of the cells per individual given the optimized number of cells per individual. The third parameter, the sample size, is defined uniquely given the other two parameters due to the budget restriction. The optimal sample size values are shown in the upper x axes (no linear scale, but matching numbers to the parameter on the lower y axis under the given budget). The horizontal line in the subplots visualizes the optimal parameter combination. The same effect sizes and expression definition as in Figure 3 was taken.

Figure S16: Parameter optimization grid

Figure S16 shows the parameter grids for the calculations done for Figure S15.

```
power.study.plot<-power.study.de
power.study.plot$totalCells<-as.factor(power.study.plot$totalCells)
power.study.plot$readDepth<-as.factor(power.study.plot$readDepth)
g.de<-ggplot(power.study.plot,aes(x=readDepth,y=totalCells,fill=`Detection power`))+
  geom_tile()+ylab("Cells per individual")+xlab("Read depth")+
  scale_fill_viridis("Detection power")+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

power.study.plot<-power.study.eqtl
power.study.plot$totalCells<-as.factor(power.study.plot$totalCells)
power.study.plot$readDepth<-as.factor(power.study.plot$readDepth)
g.eqtl<-ggplot(power.study.plot,aes(x=readDepth,y=totalCells,fill=`Detection power`))+
  geom_tile()+ylab("Cells per individual")+xlab("Read depth")+
  scale_fill_viridis("Detection power")+
  theme_bw()+
  theme(axis.title=element_text(size=10),
        axis.text=element_text(size=8),
        legend.title=element_text(size=8),
        legend.text=element_text(size=8))

g<-ggarrange(g.de,g.eqtl,ncol=2,nrow=1, labels=c("A","B"),
             common.legend = TRUE, legend="right")
print(g)
```

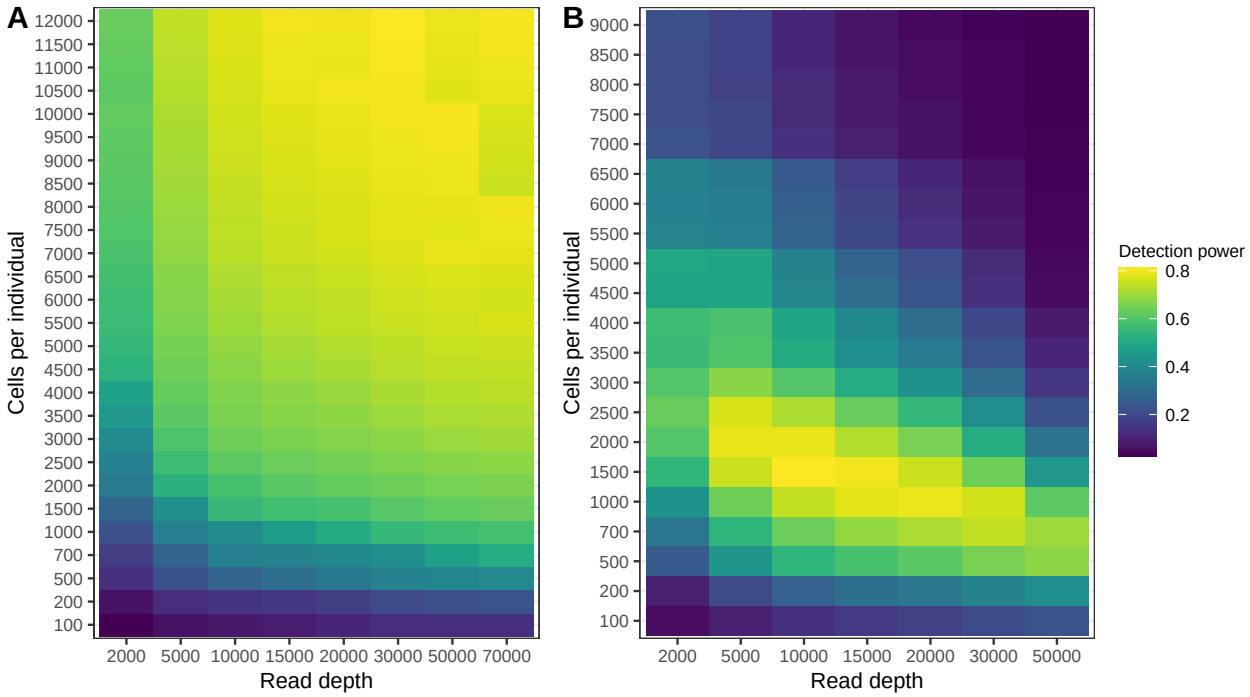

Figure S16: Parameter optimization grid. Maximizing detection power by selecting the best combination of cells per individual and read depth for A. a DE study with a budget of 10,000€ and B. a eQTL study with a budget of 30,000€, both for a cell type with a frequency of 25%. The sample size is defined uniquely given

the other two parameters due to the budget restriction. The same effect sizes and expression definition as in Figure 3 were taken.

Figure S17-S18: Variant of Main Figure 4 & 5 with FWER adjusted significance thresholds

These two figures were generated with exactly the same code as Main Figures 4 and 5, only setting the multiple testing method (MTmethod) to "FWER" instead of "FDR" for the DE power.

Figure S19: Gene curve fits for different single cell technologies

The figure shows the gene expression model for a Drop-seq lung and a Smart-seq pancreas data set, corresponding to the models shown Figure 2, S5 and S6 for a 10X Genomics PBMC data set.

```
#Load observed gene counts (precalculated)
data(precalculatedObservedGeneCounts) #observed.gene.counts

################
#Drop-seq data

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="Dropseq_count10",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-1

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits.drop,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param.drop,
    min.counts=10,
    perc.indiv=0.5,
    nGenes=24000)

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}

plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("20000","15000","10000","5000"),
                           c("complete","75","50","25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
```

```
plot.estimates$sample<-subsampled.names[plot.estimates$sample]
plot.estimates$sample<-factor(plot.estimates$sample,
                              levels=c("20000","15000","10000","5000"))

#Order variable level, first expressed.genes than estimated
plot.estimates$variable<-factor(plot.estimates$variable,
                                levels=c("expressed.genes","estimated.counts"))

g.drop<-ggplot(data=plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=col.set2)+
  scale_shape_manual("Cell type",values=seq(1,13))+
  scale_color_manual("Read depth",values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8,
        legend.direction = "vertical", legend.box = "horizontal",
        legend.position="bottom")+
  guides(linetype=FALSE,fill=FALSE,shape=guide_legend(ncol=3))

#################
#Smart-seq data

#Fit for own data and count > 10
estimates.allSamples<-observed.gene.counts[
  observed.gene.counts$evaluation=="Smartseq_count10",]

estimates.allSamples$estimated.counts<-NA
for(i in 1:nrow(estimates.allSamples)){

  sampleSize<-1

  exp.probs<-scPower::estimate.exp.prob.count.param(
    nSamples=sampleSize,
    nCellsCt=estimates.allSamples$num.cells[i]/sampleSize,
    meanCellCounts=estimates.allSamples$meanUMI[i],
    gamma.mixed.fits = scPower::gamma.mixed.fits.smart,
    ct=estimates.allSamples$cell.type[i],
    disp.fun.param = scPower::disp.fun.param.smart,
    min.counts=10,
    perc.indiv=0.5,
    nGenes=21000,
    countMethod="read")

  estimates.allSamples$estimated.counts[i]<-round(sum(exp.probs))

}
```

```
plot.estimates<-reshape2::melt(estimates.allSamples,
                               id.vars=c("run","sample","cell.type","num.cells",
                                         "meanUMI","evaluation"))

#Replace subsampling
subsampled.names<-setNames(c("500000","375000","250000","125000"),
                           c("complete","subsampling0.75","subsampling0.5",
                             "subsampling0.25"))
plot.estimates$sample<-as.character(plot.estimates$sample)
plot.estimates$sample<-subsampled.names[plot.estimates$sample]
plot.estimates$sample<-factor(plot.estimates$sample,
                              levels=c("500000","375000","250000","125000"))

#Order variable level, first expressed.genes than estimated
plot.estimates$variable<-factor(plot.estimates$variable,
                                levels=c("expressed.genes","estimated.counts"))

g.smart<-ggplot(data=plot.estimates,aes(x=num.cells,y=value))+
  geom_line(aes(color=sample,linetype=variable))+
  geom_point(aes(shape=cell.type,color=sample,fill=sample),size=2)+
  xlab("Number of cells per cell type")+
  ylab("Expressed genes")+
  scale_fill_manual(values=col.set2)+
  scale_shape_manual("Cell type",values=c(21,22,23,24,25,8,4))+
  scale_color_manual("Read depth",values=col.set2)+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        legend.title=element_text(size=6),
        legend.text=element_text(size=6),
        aspect.ratio = 0.8,
        legend.direction = "vertical", legend.box = "horizontal",
        legend.position="bottom")+
  guides(linetype=FALSE,fill=FALSE)

#####
#Combine both
g<-ggarrange(g.drop,g.smart,ncol=2,labels=c("A","B"),align="hv")

print(g)
```

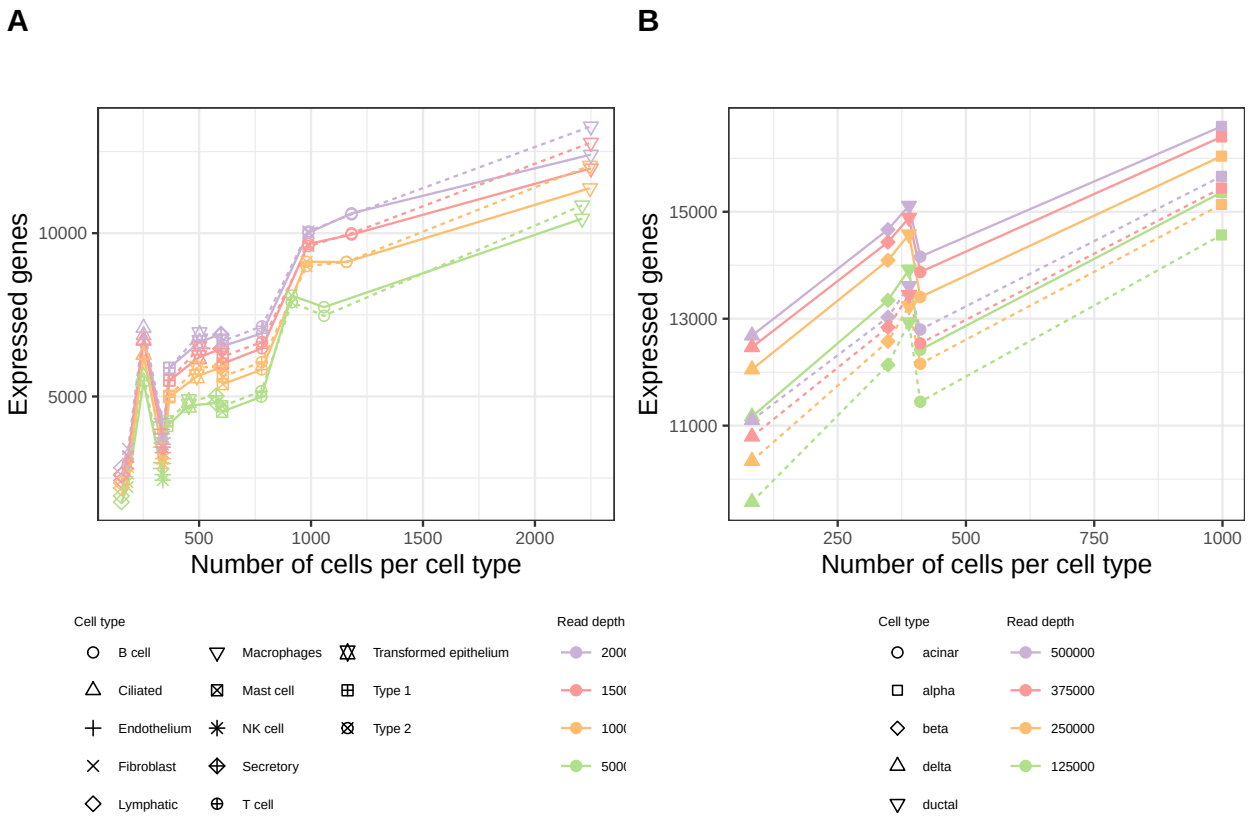

Figure S19: Gene curve fits for different single cell technologies. Plot A shows the gene curve fit for a lung data set measured with Drop-seq and plot B for a pancreas data set measured with Smart-seq2, both subsampled to different read depths. The solid lines represent the observed gene curves, the dashed lines the fitted curves. The point symbol visualizes the cell type. The definition for an expressed gene was parameterized in the following way: in Plot A, the gene needs to have UMI counts > 10 in all measured cells in total, in Plot B, read counts > 10 per kilobase transcript in all measured cells in total.

Figure S20: Comparison between powsimR and scPower for Drop-Seq and Smart-seq

Same as in Figure S14, only run for Drop-Seq and Smart-Seq data.

Figure S21: Power to detect rare cell types

Model to calculate the power to detect rare cell types. Please look into the introduction vignette, section "Part 3: Power estimation to detect a sufficent number of cells in a specific cell types" for an explanation of all parameters. The cell type frequencies are taken from BioRad (see citation in the publication), all other parameters are set to realistic values.

```
prob=0.95

#Sample size
Nind <- c(100, 150, 200, 300)

#Number of cells required
N.min.cells<-c(1,10,20,50)
```

```r
#Take cell type frequencies estimated from BioRad (see above)
cell.types.biorad<-data.frame(ct=c("B cells", "Monocytes", "NK cells",
                                   "Dendritic cells","CD4 T cells","CD8 T cells"),
                              freq=c(13,13,7,1.5,30,13)/100)


filter.ct<-c("CD4 T cells", "NK cells", "Dendritic cells", "Monocytes")
cell.type.comp<-cell.types.biorad[cell.types.biorad$ct %in% filter.ct,]

#Set frequency behind the cell type labels (add a small number to round up at 0.5)
cell.type.comp$ct<-paste0(cell.type.comp$ct,
                                   " (",round(cell.type.comp$freq*100+0.01,1),"%)")


#Arrange the cell types in the order of frequency
cell.type.comp<-cell.type.comp[order(cell.type.comp$freq),]
cell.type.comp$ct<-factor(cell.type.comp$ct,levels=cell.type.comp$ct)

#Test each possible parameter combination
parameter.combinations<-expand.grid(prob,N.min.cells,cell.type.comp$ct,Nind)
colnames(parameter.combinations)<-c("prob.cutoff","min.num.cells","ct","num.indivs")

#Merge cell types with frequencies
parameter.combinations<-merge(parameter.combinations,cell.type.comp)

parameter.combinations$result.ssize<-mapply(scPower::number.cells.detect.celltype,
                                            parameter.combinations$prob.cutoff,
                                            parameter.combinations$min.num.cells,
                                            parameter.combinations$freq,
                                            parameter.combinations$num.indivs)



#Labels samples
axis.labeller<-function(variable,value){
  return(paste(value,"individuals"))
}

g<-ggplot(parameter.combinations,aes(x=min.num.cells,y=result.ssize,col=ct))+
  geom_line()+geom_point()+facet_wrap(~num.indivs,ncol=2, labeller = axis.labeller)+
  xlab("Minimal number of cells from target cell type per individual")+
  ylab("Cells per individual")+
  scale_color_manual("Cell type",values=col.set2)+
  scale_y_log10()+
  theme_bw() +
  theme(axis.title=element_text(size=12),
        axis.text=element_text(size=8),
        aspect.ratio = 0.8,
        legend.position = "bottom")+
  guides(col=guide_legend(nrow=2,byrow=TRUE))
#> Warning: The labeller API has been updated. Labellers taking `variable` and
#> `value` arguments are now deprecated. See labellers documentation.
```

```
print(g)
```

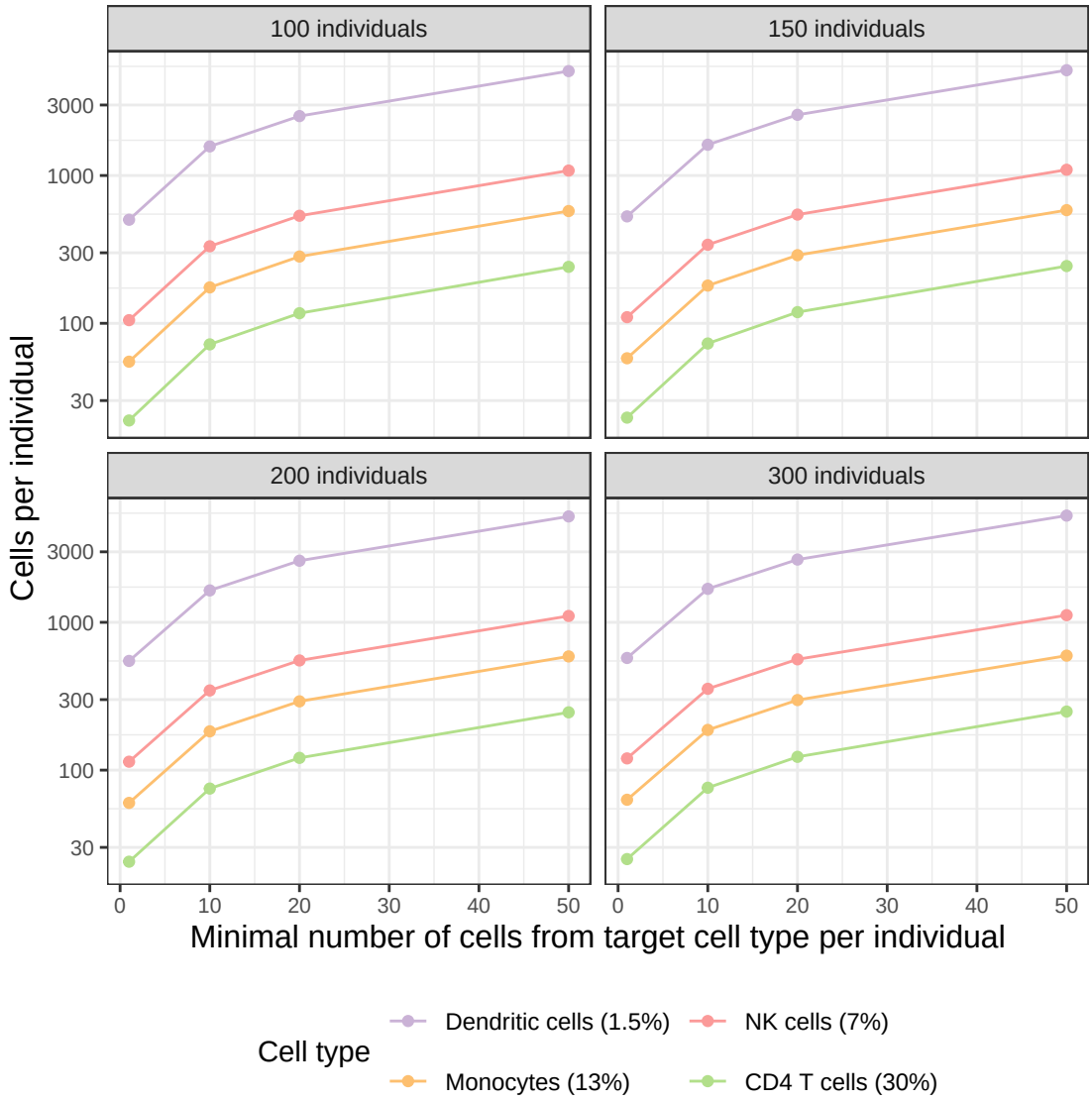

Figure S21: Power to detect rare cell types. The figure shows the required number of cells per individual (y-axis, log scale) to detect the minimal number of cells from a target cell type per individual (x-axis) with a probability of 95%. The probability depends on the total number of individuals and the frequency of the target cell type (purple, red, yellow, green). Note that the required number of cells per sample only counts "correctly measured" cells (no doublets etc), so the number is a lower bound for the required cells to be sequenced.